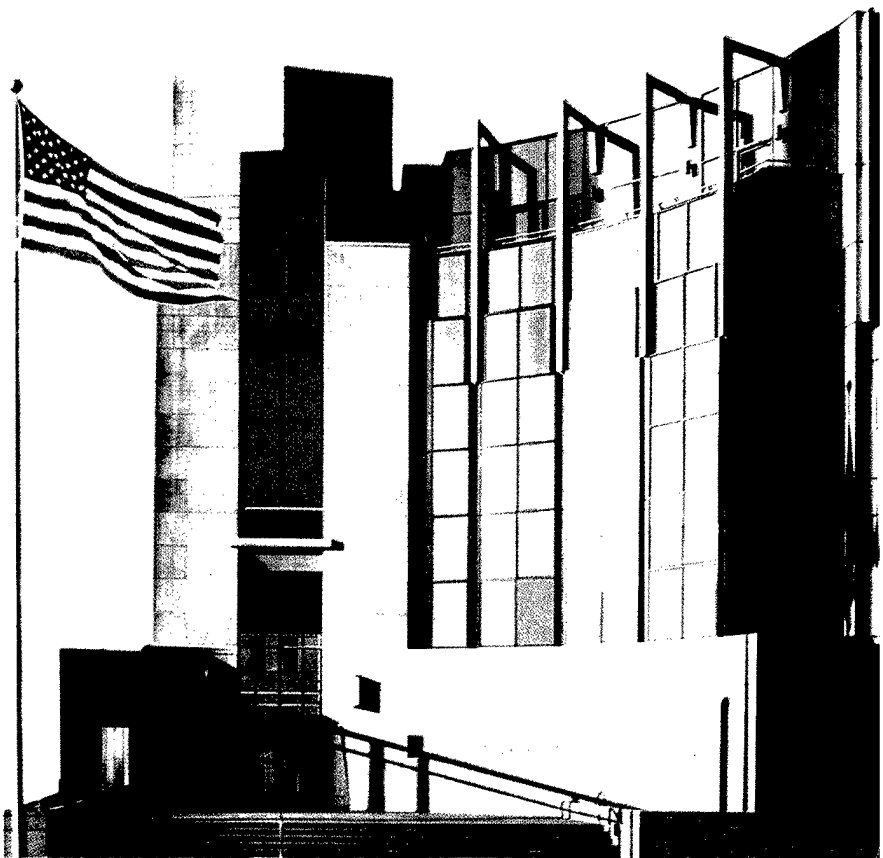
Carnegie Mellon University
Software Engineering Institute

Practical Software Measurement: Measuring for Process Management and Improvement

William A. Florac
Robert E. Park
Anita D. Carleton
April 1997

HB

GUIDEBOOK
CMU/SEI-97-HB-003



Carnegie Mellon University does not discriminate and Carnegie Mellon University is required not to discriminate in admission, employment, or administration of its programs or activities on the basis of race, color, national origin, sex or handicap in violation of Title VI of the Civil Rights Act of 1964, Title IX of the Educational Amendments of 1972 and Section 504 of the Rehabilitation Act of 1973 or other federal, state, or local laws or executive orders.

In addition, Carnegie Mellon University does not discriminate in admission, employment or administration of its programs on the basis of religion, creed, ancestry, belief, age, veteran status, sexual orientation or in violation of federal, state, or local laws or executive orders. However, in the judgment of the Carnegie Mellon Human Relations Commission, the Department of Defense policy of, "Don't ask, don't tell, don't pursue," excludes openly gay, lesbian and bisexual students from receiving ROTC scholarships or serving in the military. Nevertheless, all ROTC classes at Carnegie Mellon University are available to all students.

Inquiries concerning application of these statements should be directed to the Provost, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-6684 or the Vice President for Enrollment, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-2056.

Obtain general information about Carnegie Mellon University by calling (412) 268-2000.

Guidebook
CMU/SEI-97-HB-003
April 1997

Practical Software Measurement: Measuring for Process Management and Improvement

William A. Florac

Robert E. Park

Anita D. Carleton

Software Engineering Measurement and Analysis

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

This report was prepared for the

SEI Joint Program Office
HQ ESC/AXS
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER

Thomas R. Miller, Lt Col, USAF
SEI Joint Program Office

This work is sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number F19628-95-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 52.227-7013.

This document is available through Research Access, Inc., 800 Vinial Street, Suite C201, Pittsburgh, PA 15212. Phone: 1-800-685-6510. FAX: (412) 321-2994. RAI also maintains a World Wide Web home page. The URL is http://www.rai.com

Copies of this document are available through the National Technical Information Service (NTIS). For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161. Phone: (703) 487-4600.

This document is also available through the Defense Technical Information Center (DTIC). DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center / Attn: BRR / 8725 John J. Kingman Road / Suite 0944 / Ft. Belvoir, VA 22060-6218. Phone: (703) 767-8274 or toll-free in the U.S. — 1-800 225-3842).

It is not surprising to encounter not just a few, but many individuals who have been entrusted with continuous improvement responsibilities who cannot define an in-control process, who cannot accurately distinguish between process control and process capability, who cannot distinguish between process capability and product capability, who do not understand the basic structure of a control chart, who do not have practical knowledge of the fundamental theorem of statistical process control, and who do not understand the significance and relative importance of various signals of special causes of variation.

Robert Hoyer & Wayne Ellis, 1996

Table of Contents

List of Figures

Acknowledgments

We were assisted in much of our early work by the efforts of Andrew Huber of Data General Corporation, who worked closely with us during the year and a half in which he was a resident affiliate at the Software Engineering Institute. We thank him for the help and ideas he provided, and especially for helping us to ensure that the ideas and practices we illustrate get explained in ways that are understandable and of practical value to managers and practitioners in both industry and government.

The following people have also contributed to assembling this information, either as reviewers of drafts or as experts whose advice we sought and used as we prepared these materials. We thank them for helping make this guidebook as useful and readable as possible.

Steven Burke
Computer Sciences Corporation

David Card
Lockheed Martin

Joseph Dean
Tecolote Research, Inc.

John Gaffney
Lockheed Martin

Wolfhart Goethert
Software Engineering Institute

Jim Herbsleb
Lucent Technologies

Scott Hissam
Software Engineering Institute

Watts Humphrey
Software Engineering Institute

John McGarry
Naval Undersea Warfare Center

James Rozum
Software Engineering Institute

Norman Schneidewind
Naval Postgraduate School

Edward Weller
Bull HN Information Systems

David Zubrow
Software Engineering Institute

Preface

This guidebook is about using measurements to manage and improve software processes. It shows how quality characteristics of software products and processes can be quantified, plotted, and analyzed, so that the performance of activities that produce the products can be predicted, controlled, and guided to achieve business and technical goals. Although many of the principles and methods described in the guidebook are applicable to individual projects, the primary focus is on the enduring issues that enable organizations to improve not just today's performance, but the long-term success and profitability of their operations.

If you are a software manager or practitioner who has responsibilities for quality or performance that extend beyond just the project of the day, and if you are experienced in defining, collecting, and using measurements to plan and manage projects, then this guidebook is for you. It will start you on the road to using measurement data to control and improve process performance. Not only will the discussions here introduce you to important concepts, but they will also point you to additional things that you will ultimately want to know to fully implement and use the power of quantitative information.

On the other hand, if your organization does not yet have basic measurement processes in place, you should make establishing measures for planning and managing projects your first priority. Handbooks such as *Practical Software Measurement: A Guide to Objective Program Insight* [JLC 96] and *Goal-Driven Software Measurement* [Park 96a] make excellent starting points, as do the examples and advice found in books by people such as Watts Humphrey and Robert Grady [Humphrey 89, Grady 87, Grady 92].

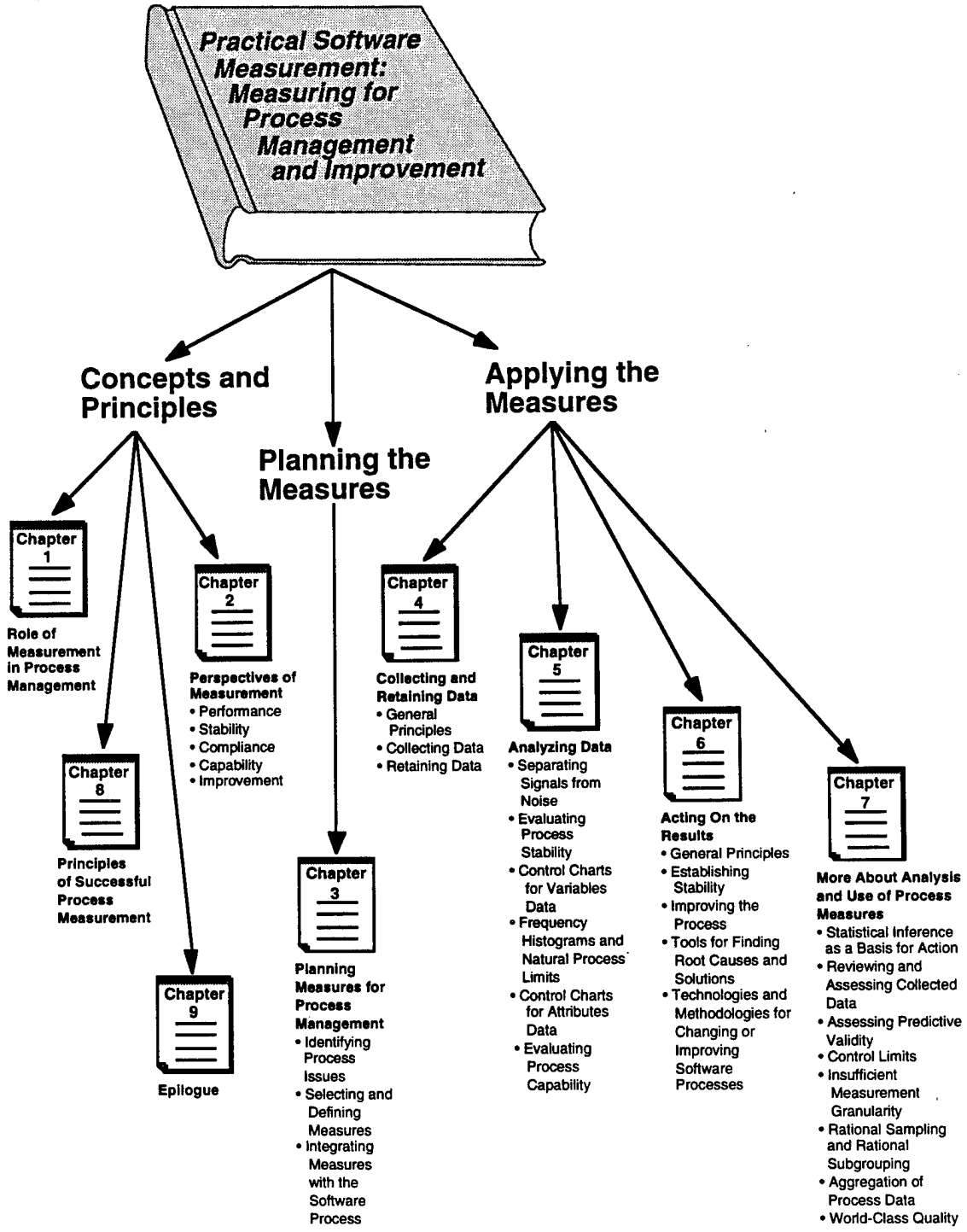
The discussions in this guidebook will not attempt to teach you all that you need to know about using measurements to manage and improve software processes. Nor will they attempt to teach you everything there is to know about using control charts and other empirical tools of quality and process improvement. The mechanics associated with these tools are explained adequately elsewhere, and we provide numerous pointers to books that provide instruction on these subjects. Instead, this guidebook focuses on things that we believe to be more important—the concepts and issues that lie behind the empirical (statistical) methods, and the steps associated with effectively implementing and using the methods to manage and improve software processes.

This document, then, is a guidebook in the sense that it describes and illustrates the best practices that we can currently identify for gathering and using quantitative data to help manage and improve software processes. Everything in this guidebook has its roots in experience: often that of the authors, at times that of others. Much of the experience comes from software settings, while other lessons have been drawn from more physically oriented environments. Although some of the numerical examples are composed rather than factual, we have tried to ensure that they represent reasonable extensions of practices that have been demonstrated to be successful, if not in software settings, at least in other environments.

Although this guidebook emphasizes the principles of statistical quality control (SQC), we recognize that these principles have received little use as of yet in software organizations. Nevertheless, we refuse to let the current state of software measurement stop us from promoting their use. The benefits of the empirical methods associated with SQC have been so evident in other development, production, and service activities that it would be foolish to ignore their potential for improving software products and services.

We recognize that there is much more to be said about the use of measurements and statistical methods for managing and improving processes than this guidebook now addresses. We also recognize that, in selecting the principal topics, we may be guilty of overemphasizing control charts while underemphasizing the fact-finding and improvement activities that must take place once problems and improvement opportunities are detected. The latter subjects are all grist for future versions of the guidebook. For now, though, the focus is on the initial steps of acquiring and using quantitative information in ways that can help you to reliably identify the problems and opportunities present in the processes you operate, so that you can use the results to guide your management and improvement actions. We believe that the guidebook provides ideas and methods that you will find useful, and we hope that it will encourage you and others to begin applying the concepts that it explains and illustrates.

Guidebook Roadmap



Measuring for Process Management and Improvement

Abstract: This guidebook shows how well-established principles and methods for evaluating and controlling process performance can be applied in software settings to help achieve an organization's business and technical goals. Although the concepts that are illustrated are often applicable to individual projects, the primary focus is on the enduring issues that enable organizations to improve not just today's performance, but the long-term success and profitability of their business and technical endeavors.

1 The Role of Measurement in Process Management

> *The central problem of management in all its aspects, including planning, procurement, manufacturing, research, sales, personnel, accounting, and law, is to understand better the meaning of variation, and to extract the information contained in variation.*
>
> *Lloyd S. Nelson, as quoted by Deming [Deming 86a]*

Every organization asks the question, "Are we achieving the results we desire?" This gets couched in many ways, such as, "Are we meeting our business objectives? Are our customers satisfied with our product and services? Are we earning a fair return on our investment? Can we reduce the cost of producing the product or service? How can we improve the response to our customers' needs or increase the functionality of our products? How can we improve our competitive position? Are we achieving the growth required for survival?"

The answers to questions like these are rooted in an organization's basic makeup, its overall philosophy, its people, its management, and the facilities and operational processes used by the organization to deliver its products or services. It is the operational processes that concern us in this guidebook.

1.1 Why Measure?

> *Without the right information, you're just another person with an opinion.*
>
> *— Tracy O'Rourke, CEO of Allen-Bradley*

Advances in technology are continually increasing the demand for software that is larger, more robust, and more reliable over ever-widening ranges of application. The demands on

software management are increasing correspondingly. Software developers and maintainers—managers and technical staff alike—are repeatedly confronted with new technologies, more competitive markets, increased competition for experienced personnel, and demands for faster responsiveness [Card 95]. At the same time, they continue to be concerned about open-ended requirements, uncontrolled changes, insufficient testing, inadequate training, arbitrary schedules, insufficient funding, and issues related to standards, product reliability, and product suitability.

Software measurement by itself cannot solve these problems, but it can clarify and focus your understanding of them. Moreover, when done properly, sequential measurements of quality attributes of products and processes can provide an effective foundation for initiating and managing process improvement activities.

The success of any software organization is contingent on being able to make predictions and commitments relative to the products it produces. Effective measurement processes help software groups succeed by enabling them to understand their capabilities, so that they can develop achievable plans for producing and delivering products and services. Measurements also enable people to detect trends and to anticipate problems, thus providing better control of costs, reducing risks, improving quality, and ensuring that business objectives are achieved.

In short, measurement methods that identify important events and trends and that effectively separate signals from noise are invaluable in guiding software organizations to informed decisions.

1.2 Process Measures Are Driven by Goals and Issues

For measurement activities to be cost effective, they must be designed and targeted to support the business goals of your organization and provide effective, economical information for decision making. This is not as simple and straightforward as it may sound. One of the dangers in enterprises as complex as software development and support is that there are potentially so many things to measure that we are easily overwhelmed by opportunities [Park 96a].

Experience has taught us that we must identify the critical factors that determine whether or not we will be successful in meeting our goals. These critical factors are often associated with issues. Issues, in turn, relate to risks that threaten our ability to meet goals, responsibilities, or commitments. Goals and issues serve to identify and focus the measurements needed to quantify the status and performance of software processes.

To help address business goals, we can usefully view software management functions as falling into three broad classes—*project management*, *process management*, and *product engineering*. These management functions address different concerns, each with its own objectives and issues. For example:

- **project management**. The objectives of software *project management* are to set and meet achievable commitments regarding cost, schedule, quality, and function delivered—as they apply to individual development or maintenance projects. The key management issues are those that appear to jeopardize these commitments. Software project management is interested primarily in creating achievable plans and in tracking the status and progress of its products relative to its plans and commitments.
- **process management**. The objectives of *process management* are to ensure that the processes within the organization are performing as expected, to ensure that defined processes are being followed, and to make improvements to the processes so as to meet business objectives (e.g., lowering risks associated with commitments and improving the ability to produce quality products).
- **product engineering**. The objectives of *product engineering* are to ensure customer acceptance of and satisfaction with the product. The issues of greatest concern relate primarily to the physical and dynamic attributes of the product—architecture, producibility, reliability, usability, responsiveness, stability, performance, and so forth. Information about these attributes and customer satisfaction is important to assessing the attainment of product engineering goals.

The kinds of data that are needed to achieve these objectives can differ significantly across these three management functions. Moreover, there are many interactions among the functions in most software organizations. In many cases, these interactions lead to conflicting demands that must be managed. In addition, various organizational entities—corporate, division, program, staff, project, and functional groups—often have different goals, together with differing issues, perspectives, and interests, even when sharing the same overall business goals. This frequently leads to differing measurement needs and priorities. But whatever the priorities, the business goals, objectives, strategies, and plans for all organizations are formed around the fundamental objectives of

- providing competitive products or services in terms of functionality, time to market, quality, and cost
- meeting commitments to customers with respect to products and services

Success in meeting commitments while achieving or exceeding goals for functionality, quality, cost, and time to market implies a need to ensure that commitments are achievable. This in turn implies a need to predict outcomes and evaluate risks.

The processes that software organizations use to produce products and services have critical roles in the execution of strategies and plans aimed at these objectives. Organizations that can control their processes are able to predict the characteristics of their products and services, predict their costs and schedules, and improve the effectiveness, efficiency, and profitability of their business and technical operations.

For example, if our business goals are based on function, cost, time to market, and quality, we can identify both project and process issues (concerns) that relate to achieving these goals. Process performance can then be quantified by measuring attributes of products produced by our processes as well as by measuring process attributes directly. Figure 1-1 lists some typical business goals that often concern us and relates these goals to corresponding project and process issues and examples of attributes that can be measured to assess the performance of a process with respect to these issues.

Business Goals	Project Issues	Process Issues	Measurable Product and Process Attributes
increase function	product growth product stability	product conformance	number of requirements product size product complexity rates of change % nonconforming
reduce cost	budgets expenditure rates	efficiency productivity rework	product size product complexity effort number of changes requirements stability
reduce the time to market	schedule progress	production rate responsiveness	elapsed time, normalized for product characteristics
improve product quality	product performance product correctness product reliability	predictability problem recognition root cause analysis	number of defects introduced effectiveness of defect detection activities

Figure 1-1: Business Goals, Project and Process Issues, and Related Measurable Attributes

Measurements of attributes like those shown in column four of Figure 1-1 are important not just because they can be used to describe products and processes, but because they can be used to control the processes that produce the products, thus making future process performance predictable. Measurements of product and process attributes can also be used to quantify process performance and guide us in making process improvements. This in turn helps keep our operations competitive and profitable.

In addition to product and process attributes like those listed in column four of Figure 1-1, there are two properties of the process itself that are important to successfully achieving

business goals: *process stability* and *process capability.* These properties are orthogonal to all measures of process and product attributes.

Process stability, as we shall see shortly, lies at the heart of all process management and process improvement efforts. It is what enables us to act as if results are repeatable. Process stability lets us predict future performance and prepare achievable plans. It also provides a basis for separating signal from noise, so that departures from stability can be recognized and become the basis for effective control and improvement actions.

If a process is in statistical control, and if a sufficiently large proportion of the results fall within the specification limits, the process is termed *capable.* Thus a capable process is a stable process whose performance satisfies customer requirements.

Stability and capability will be described further in Chapter 2 and illustrated in greater detail in Chapter 5. For now, we simply point out that assessments of process stability and process capability are obtained by measuring attributes of product and process quality and analyzing the variability and central tendency in the measured results. The measures that we use are the same ones that we use when quantifying business-related issues like those listed in Figure 1-1. We also use the same measures, together with data that characterize the resources used by the processes, to quantify and analyze issues associated with process compliance and process improvement.

Because of the multiplicity of goals, issues, processes, and perspectives within any organization, it makes little sense to attempt to lay out detailed instructions for measuring all conceivable product and process attributes. Instead, this guidebook focuses on the issues of process management and on assembling guidelines and advice for intelligently using measures to control and improve the processes that exist within software organizations.

1.3 What Is a Software Process?

The term "process" means different things to different people. So it is important to clearly define what we mean when we use the word, especially in the context of a software development or support environment.

> *A process can be defined as the logical organization of people, materials, energy, equipment, and procedures into work activities designed to produce a specified end result.*
>
> *Gabriel Pall, 1987*

This definition is illustrated in Figure 1-2. It differs from the definitions of *process* and *software process* given in the Capability Maturity Model[sm] (CMM[sm]) for Software[1] in that it

[1]CMM and Capability Maturity Model are service marks of Carnegie Mellon University.

includes people, materials, energy, and equipment within the scope of *process*. Version 1.1 of the CMM, by way of contrast, views a process as a "sequence of steps" performed by people with the aid of tools and equipment to transform raw material into a product [Paulk 93b, Paulk 95].

Including people, materials, energy, and tools within the concept of process becomes important when we begin to apply the principles of statistical process control to improve process performance and process capability. Here we use measures of variability to identify opportunities for improving the quality of products and the capability of processes. When searching for causes of unusual variation, it would be a mistake to exclude people, materials, energy, and tools from the scope of the investigation. This view of "process," as enunciated by Pall, has been at the very heart of statistical process control since its founding in the 1920s [Shewhart 31, Western Electric 58, Juran 88].

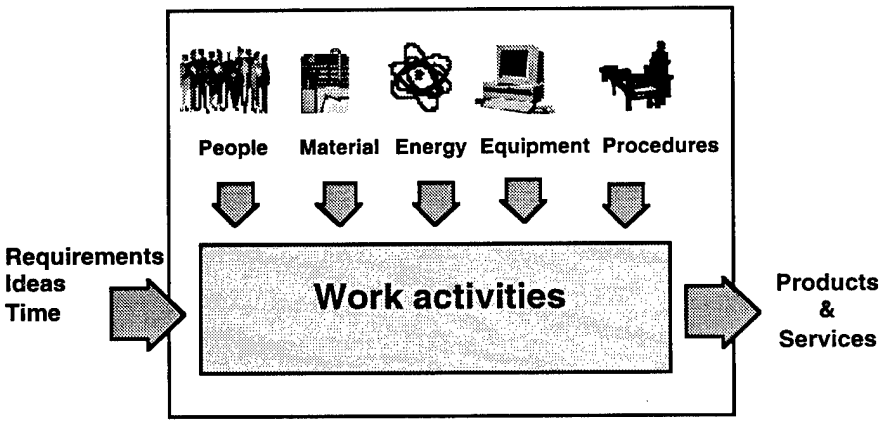


Figure 1-2: Definition of Process

In keeping with the broad view of process, we use the term *software process* in this guidebook to refer not just to an organization's overall software process, but to any process or subprocess used by a software project or organization. In fact, a good case can be made that it is only at subprocess levels that true process management and improvement can take place. Thus, readers should view the concept of software process as applying to any identifiable activity that is undertaken to produce or support a software product or service. This includes planning, estimating, designing, coding, testing, inspecting, reviewing, measuring, and controlling, as well as the subtasks and activities that comprise these undertakings.

1.4 What Is Software Process Management?

Software process management is about successfully managing the work processes associated with developing, maintaining, and supporting software products and software-intensive systems. By successful management, we mean that the products and services produced by the processes conform fully to both internal and external customer requirements, and that they meet the business objectives of the organization responsible for producing the products.

The concept of process management is founded on the principles of statistical process control. These principles hold that by establishing and sustaining stable levels of variability,

processes will yield predictable results. We can then say that the processes are *under statistical control.* This was first enunciated by Walter Shewhart:

> *A phenomenon will be said to be controlled when, through the use of past experience, we can predict, at least within limits, how the phenomenon may be expected to vary in the future.*
>
> *Walter A. Shewhart, 1931*

Predictable results should not be construed to mean identical results. Results always vary; but when a process is under statistical control, they will vary within predictable limits. If the results of a process vary unexpectedly—whether randomly or systematically—the process is not under control, and some of the observed results will have assignable causes. These causes must be identified and corrected before stability and predictability can be achieved.

Controlled processes are stable processes, and stable processes enable you to predict results. This in turn enables you to prepare achievable plans, meet cost estimates and scheduling commitments, and deliver required product functionality and quality with acceptable and reasonable consistency. If a controlled process is not capable of meeting customer requirements or other business objectives, the process must be improved or retargeted.

At the individual level then, the objective of software process management is to ensure that the processes you operate or supervise are predictable, meet customer needs, and (where appropriate) are continually being improved. From the larger, organizational perspective, the objective of process management is to ensure that the same holds true for every process within the organization.

1.5 Responsibilities and Objectives of Software Process Management

In this section, we identify four key responsibilities of software process management and show how they relate to the responsibilities of project management. Our discussion of the activities and issues associated with the responsibilities will be neither all encompassing nor definitive, but it will provide a starting point from which measurement guidelines can be developed in something more substantial than just abstract terms.

There are four responsibilities that are central to process management:

- Define the process.
- Measure the process.
- Control the process (ensure that variability is stable so that results are predictable).
- Improve the process.

In contrast, *project* management is responsible for seeing that a software product is developed according to a plan and that the plan is feasible. Thus, the principal objectives of project management are to set and meet achievable commitments with respect to cost, schedule, function delivered, and time to market. But without the underlying process management activities, project managers assume significant risk in both setting and meeting these commitments.

The four responsibilities of process management are shown as boxes in Figure 1-3. Execution of the process is depicted with a different shape because execution is not a process management responsibility. Rather, it is an inherent responsibility of project management, whether performed by a software developer or a software maintainer. People responsible for process management may have project management responsibilities as well, and project managers may implicitly assume process management responsibility for processes that they define and use.

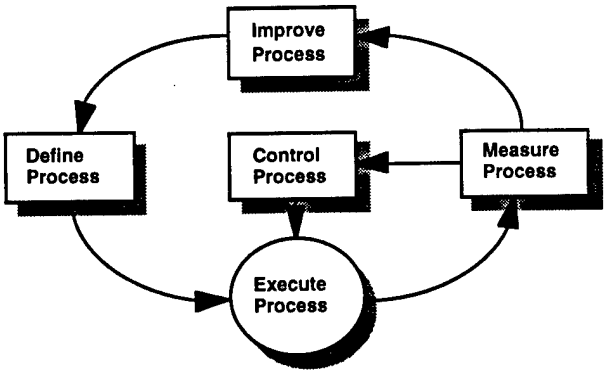


Figure 1-3: The Four Key Responsibilities of Process Management

A concept of process ownership is implicit in Figure 1-3. Responsibility for this ownership lies with the function of process management. This is especially so whenever processes cross organizational boundaries or involve multiple organizational entities. Process ownership includes responsibilities for process design, for establishing and implementing mechanisms for measuring the process, and for taking corrective action where necessary [Pall 87].

The following paragraphs outline the four key responsibilities of process management.

Define the Process

Defining a software process creates the disciplined and structured environment required for controlling and improving the process. Management's responsibility to define each process inherently includes responsibilities for implementing and sustaining the process. The key objectives associated with defining, implementing, and sustaining are

- Design processes that can meet or support business and technical objectives.
- Identify and define the issues, models, and measures that relate to the performance of the processes.
- Provide the infrastructures (the set of methods, people, and practices) that are needed to support software activities.
- Ensure that the software organization has the ability to execute and sustain the processes (skills, training, tools, facilities, and funds).

Measure the Process

Measurements are the basis for detecting deviations from acceptable performance. They are also the basis for identifying opportunities for process improvement. The key objectives of process measurement are to

- Collect data that measure the performance of each process.
- Analyze the performance of each process.
- Retain and use the data to
  - assess process stability and capability
  - interpret the results of observations and analyses
  - predict future costs and performance
  - provide baselines and benchmarks
  - plot trends
  - identify opportunities for improvement

Control the Process

> *An adequate science of control for management should take into account the fact that measurements of phenomena in both social and natural science for the most part obey neither deterministic nor statistical laws, until assignable causes of variability have been found and removed.*
>
> *Walter A. Shewhart, 1943*

Controlling a process means keeping the process within its normal (inherent) performance boundaries—that is, making the process behave consistently. This involves

- **measurement**: obtaining information about process performance
- **detection**: analyzing the information to identify variations in the process that are due to assignable causes

- **correction**: taking steps to remove variation due to assignable causes from the process and to remove the results of process drift from the product [Pall 87]

To say this another way, the three key actions needed to establish and maintain control of a software process are as follows:

1. Determine whether or not the process is under control (i.e., stable with respect to the inherent variability of measured performance).
2. Identify performance variations that are caused by process anomalies (assignable causes).
3. Eliminate the sources of assignable causes so as to stabilize the process.

Once a process is under control, sustaining activities must be undertaken to forestall the effects of entropy. Without sustaining activities, processes can easily fall victim to the forces of ad hoc change or disuse and deteriorate to out-of-control states. This requires reinforcing the use of defined processes through continuing management oversight, benchmarking, and process assessments.

Improve the Process

Even though a process may be defined and under control, it may not be capable of producing products that meet customer needs or organizational objectives. For most organizations, processes must be technologically competitive, adaptable, and timely, and they must produce products that consistently meet customer and business needs. Moreover, resources must be available and capable of performing and supporting the processes. Processes can be improved by making changes that improve their existing capabilities or by replacing existing subprocesses with others that are more effective or efficient. In either case, the process improvement objectives of an organization are to

- Understand the characteristics of existing processes and the factors that affect process capability.
- Plan, justify, and implement actions that will modify the processes so as to better meet business needs.
- Assess the impacts and benefits gained, and compare these to the costs of changes made to the processes.

1.6 The Relationship of Process Management to Project Management

Figure 1-4 extends the process management model depicted in Figure 1-3. It illustrates the relationships between process management and project management responsibilities and shows how measurement activities relate to these responsibilities.

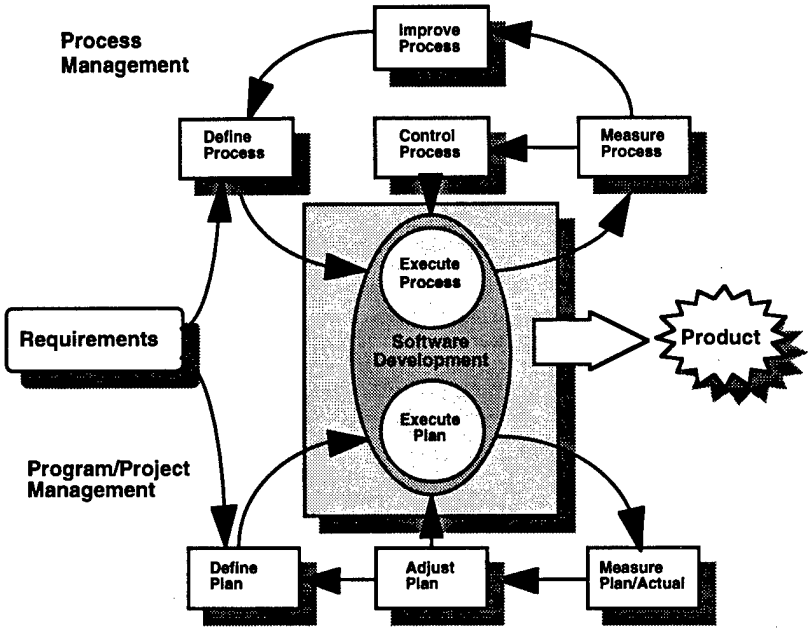


Figure 1-4: The Relationship Between Process Management and Project Management

As Figure 1-4 indicates, a software project team produces products based on three primary ingredients: product requirements, a project plan, and a defined software process. It follows, then, that *project* management's use of measurement data will be guided by its needs to

- identify and characterize requirements
- prepare a plan that lays out achievable objectives
- implement the plan
- track the status and progress of work performed with respect to the goals laid out in the project plan[2]

Process management, on the other hand, uses the same data and other related measurements to control and improve the software process itself. This means that

[2]Examples of measurement activities associated with software project management are described in *Practical Software Measurement: A Guide to Objective Program Insight* [JLC 96].

organizations can use a common framework for establishing and sustaining the measurement activities that provide the data for both management functions.

1.7 Integrating Measurement with Software Process Management

Figure 1-5 extends the basic process management model again, but in a way that is different from that in Figure 1-4. It shows the relationships between process management responsibilities and the planning and applying activities of the measurement process. (The measurement activities are highlighted.) The interactions between measurement and process management are summarized in the paragraphs that follow the figure.

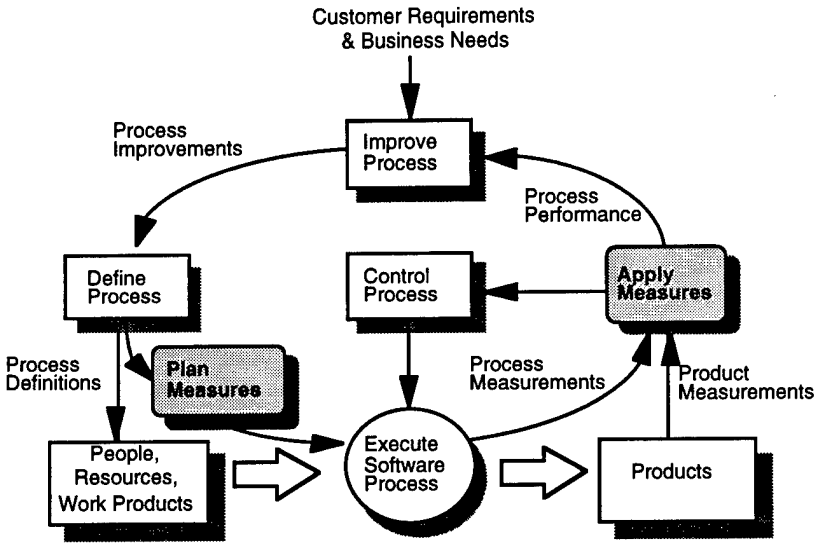


Figure 1-5: Measurement Activities and Their Relationship to the Key Responsibilities of Process Management

1. **Define the process.** A process is an organized combination of people, materials, energy, equipment, and procedures engaged in producing a specified end result—often a product or service. Prior to selecting and implementing measures, each contributing element of the process must be identified, and a thorough understanding of the process operation and objectives must be attained by those engaged in process management. Data-flow diagrams and control-flow diagrams can be useful tools for documenting and communicating understandable and usable (i.e., operational) definitions.

2. **Plan the measures.** Measurement planning is based on an understanding of the defined (or implicit) software process. Here the product-, process-, and resource-related issues and attributes are identified; measures of product and process quality are selected and defined; and provisions for

collecting and using the measurements to assess and track process performance are integrated into the software process.

3. **Execute the software process.** Processes are executed by the software organization. The product, process, and resource attributes that were identified are measured during and at the completion of each software process.

4. **Apply the measures.** Applying measures puts to use the measurements that are obtained while executing the software process. Data from the software process and from products produced by the process are collected, retained, and analyzed so that they can be used to control and improve the process.

5. **Control the process.** If measurements of product or performance attributes indicate that the process varies in unexpected or unpredictable ways, actions must be taken to remove assignable causes, stabilize the variability, and (if appropriate) return the process to its natural level of performance.

6. **Improve the process.** Once measurements indicate that all variability in a process comes from a constant system of chance causes (i.e., only natural or inherent variation exists), process performance data can be relied on and used to guide actions aimed at changing the level of performance. Improvement actions whose benefits are subsequently validated by measurement can then be used to update and evolve the process definition.

2 The Perspectives of Process Measurement

> *...the object of control is to enable us to do what we want to do within economic limits.*
>
> *Walter A. Shewhart, 1931*

> *A process that is out-of-control does not exist as a well-defined entity.*
>
> *Donald J. Wheeler and David S. Chambers, 1992*

The purpose of this chapter is to outline five perspectives that are central to process measurement:

- performance
- stability
- compliance
- capability
- improvement and investment

We will examine these perspectives in the context of the responsibilities and objectives of process management described in Chapter 1. The discussions here lay foundations for concepts introduced subsequently in Chapters 3 through 7, where we look at planning measurement activities and applying the results to manage and improve software process performance.

In Chapter 1 we described the four key responsibilities of process management—*define, measure, control,* and *improve.* Although that discussion implied a sequence that begins with *define*, in practice it is helpful to know what and why we are defining before we set about constructing and implementing that portion of a software process that deals with measurement. To explore these issues, we direct our attention to the *Control Process* responsibility that is highlighted in Figure 2-1. This will help us introduce several important perspectives of process measurement very quickly.

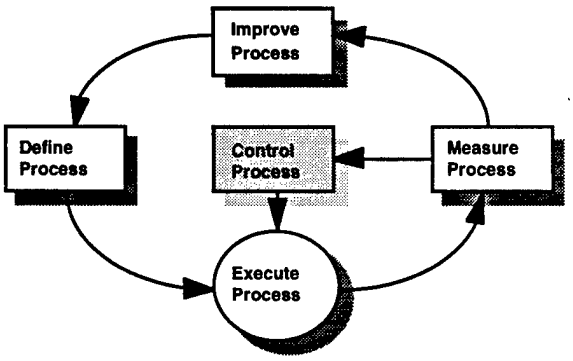


Figure 2-1: The *Control-Process* Responsibility

Controlling a process means making it behave the way we want it to. Control enables organizations to do two things: predict results and produce products that have characteristics desired by customers. With control, we can commit to dates when products will be delivered and live up to such commitments.

Controlling a process is essential to producing products that customers can afford and that we can profit by. In short, control is central to achieving the business goals and objectives that make a software organization successful.

The first step in controlling a process is to find out what the process is doing now. All processes are designed to produce results. The products and services they deliver and the ways they deliver them have measurable attributes that can be observed to describe the quality, quantity, cost, and timeliness of the results produced. If we know the current values of these attributes, and if a process is not delivering the qualities we desire, we will have reference points to start from when introducing and validating process adjustments and improvements.

So our first concern when measuring for process management and improvement is to understand the existing *performance* of the processes we use—what are they producing now? Knowing how a process is performing will enable us to assess the repeatability of the process and whether or not it is meeting its internal and external needs. Notice that we said "how," not "how well." When we measure process performance, our purpose is not to be judgmental, but simply to get the facts. Once the facts are in hand and we know the current levels and variabilities of the values that are measured, we can proceed to evaluating the information from other perspectives. Our first concern, then, is as follows:

- **Performance**—What is the process producing now with respect to measurable attributes of quality, quantity, cost, and time?

Measures of process performance quantify and make visible the ability of a process to deliver products with the qualities, timeliness, and costs that customers and businesses require. When measurements of process performance vary erratically and unpredictably over time, the process is not in control. To attain control, we must ensure first that we have a process whose variability is stable, for without stability we cannot predict results. So another important property associated with any process is that of *process stability.*

- **Stability**—Is the process that we are managing behaving predictably?

How do we know if a process is stable? We must first define what we mean by stable (see Sections 1.4 and 2.2), and then we must find ways of measuring appropriate process and product attributes to determine if stability has been achieved. Thus, as you may have already surmised, measurement lies at the heart of quantitatively understanding any process management issue.

If process performance is erratic and unpredictable, we must take action to stabilize that process. Stability of a process depends on support for and faithful operation of the process. Three questions that should concern people responsible for processes are

- Is the process supported such that it will be stable if operated according to the definition?

- Is the process, as defined, being executed faithfully?
- Is the organization fit to execute the process?

Questions of this sort address the issue of *process compliance.*

- **Compliance**—Are the processes sufficiently supported? Are they faithfully executed? Is the organization fit to execute the process?

Having a stable and compliant process does not mean that process performance is satisfactory. The process must also be *capable.* Capable means that variations in the characteristics of the product and in the operational performance of the process, when measured over time, fall within the ranges required for business success. Measures of process capability relate the performance of the process to the specifications that the product or process must satisfy.

- **Capability**—Is the process capable of delivering products that meet requirements? Does the performance of the process meet the business needs of the organization?

If a software process is not capable of consistently meeting product requirements and business needs, or if an organization is to satisfy ever-increasing demands for higher quality, robustness, complexity, and market responsiveness while moving to new technologies and improving its competitive position, people in the organization will be faced with the need to continually improve process performance. Understanding the capability of the subprocesses that make up each software process is the first step in making progress towards *process improvement.*

- **Improvement**—What can we do to improve the performance of the process? What would enable us to reduce variability? What would let us move the mean to a more profitable level? How do we know that the changes we have introduced are working?

The next four sections discuss the topics of process performance, stability, compliance, and capability in more detail. Section 2.5 then introduces the topic of process improvement. How measures can be used to address these issues is illustrated in Chapters 3 through 7.

2.1 Performance

What do we mean by process performance?

Process performance refers to the characteristic values we see when measuring attributes of products and services that come from a process. Process performance can be described in two ways: by measuring attributes of products that the process produces and by measuring attributes of the process itself. Histories of these values describe how the

process has performed in the past. Under the right conditions, the histories give reference points for judging what the process is doing now and might do in the future.

Examples of measurable product attributes include things such as function, size, execution speed, module strength, and profiles of statement types. They also include any quality attributes (such as ease of use and reliability) that would make the product more or less desirable to a user or customer. Examples of process attributes, on the other hand, include things like the amount of effort expended, the clock time or computer time required to perform a process or task, the sizes and durations of work flows and backlogs, the characteristics and quantities of resources used, and the numbers, types, and sources of defects detected, repaired, or reported.

Product and process measures may well be the same measures. Measurements of product attributes such as McCabe complexity, average module size, and degree of code reuse that are tracked over time and used to investigate trends in process performance are just a few examples.

We may also seek reasons for variations in process performance by measuring attributes of the resources or environments that support the process—for example, the experience and training levels of the software engineers or the amount of computer time or memory available.

In each case, the key to measuring process performance is to choose attributes and measures that are relevant to the particular process and issues under study. We do this by selecting measures that not only reflect the intended purpose of the process, but also address the issues being considered.

To measure process performance, we measure as many product-quality and process-performance attributes as needed, and we do this at several points in time to obtain sequential records of their behavior. It is these sequential records that are the basis for establishing statistical control and, hence, for assessing the stability and capability of the process. The properties of stability and capability are illustrated in Sections 2.2 and 2.4 of this chapter.

The choices of attributes to measure will change from process to process and from issue to issue. This is where your knowledge of your organization's processes comes into play. We offer the guidelines in Figure 2-2 as a check against your selections.

For performance data to be well defined, as required by the guidelines in Figure 2-2, they must satisfy three criteria:

- **communication**. Will the methods used to define measures or describe measured values allow others to know precisely what has been measured and what has been included in and excluded from aggregated results? Moreover, will every user of the data know how the data were collected, so that they can interpret the results correctly?

Measures used to characterize process performance should

- relate closely to the issue under study. These are usually issues of quality, resource consumption, or elapsed time.
- have high information content. Pick measures of product or process qualities that are sensitive to as many facets of process results as possible.
- pass a reality test. Does the measure really reflect the degree to which the process achieves results that are important?
- permit easy and economical collection of data.
- permit consistently collected, well-defined data.
- show measurable variation. A number that doesn't change doesn't provide any information about the process.
- as a set, have diagnostic value. They should be able to help you identify not only that something unusual has happened, but what might be causing it.

Figure 2-2: Guidelines for Selecting Product and Process Measures

- **repeatability**. Would someone else be able to repeat the measurements and get the same results?
- **traceability**. Are the origins of the data identified in terms of time, sequence, activity, product, status, environment, measurement tools used, and collecting agent?

The traceability requirement is especially important to assessing and improving process performance. Because measures of performance can signal process instabilities, it is important that the context and circumstances of the measurement be recorded. This will help in identifying assignable causes of the instabilities.

For example, suppose that a software problem report is prepared whenever a defect is encountered by a defect-finding activity. To find and fix the defect, programmers and analysts must know as much as possible about the circumstances of the encounter—how, when, where, what happened, under what conditions, and so forth. So it is helpful if this information is recorded on the problem report.

For the same reason, measurements of process performance should always be accompanied by similar contextual information. When the data show process instabilities, the people who evaluate the data will look for incidents or events that are not part of the

normal process in order to identify assignable causes. Contextual information, together with data gathered from measures of process compliance, will help them identify these causes.

Wheeler and Chambers, on page 112 of their book *Understanding Statistical Process Control*, discuss the need for contextual information [Wheeler 92]. They give examples of the kinds of questions that must be answered if performance data are to be interpreted correctly. Some of these questions are

- What do the individual values represent? What are these numbers?
- How are the values obtained? Who obtains them? How often? At which location? By what method? With what instrumentation?
- What sources of variation are present in these data?
- How are these data organized into subgroups? Which sources of variation occur *within* the subgroups? Which sources of variation occur *between* the subgroups?
- How should such data behave? Are there natural barriers within the range of observed values?

When you collect product and process measures, you should always be prepared to answer these kinds of questions. The answers will help point you (correctly) to assignable causes.

2.2 Stability

> *It is only in the state of statistical control that statistical theory provides, with a high degree of belief, prediction of performance in the immediate future.*
>
> *W. Edwards Deming, 1993*

> *...the necessary and sufficient condition for statistical control is that the causes of an event satisfy the law of large numbers as do those of a constant system of chance causes.*
>
> *Walter A. Shewhart, 1931*

> *...any unknown cause of a phenomenon will be termed a chance cause.*
>
> *Walter A. Shewhart, 1931*

Process stability is considered by many to be at the core of process management. It is central to each organization's ability to produce products according to plan and to improve processes so as to produce better and more competitive products.

What is meant by stability? To answer this question, we must first understand that almost all characteristics of processes and products display variation when measured over time. This variation has two sources:

- phenomena that are natural and inherent to the process and whose results are common to all measurements of a given attribute
- variations that have assignable causes that could have been prevented

In equation form, the concept is

[total variation] = [common cause variation] + [assignable cause variation]

Common cause variation is the normal variation of the process. It exists because of normal interactions among the components of the process (people, machines, material, environment, and methods). Common cause variation is characterized by a stable and consistent pattern over time, as illustrated in Figure 2-3. The results of common cause variation are thus random, but they vary within predictable bounds. When a process is stable, the random variations that we see all come from a constant system of chance causes. The variation is predictable, and unexpected results are extremely rare.

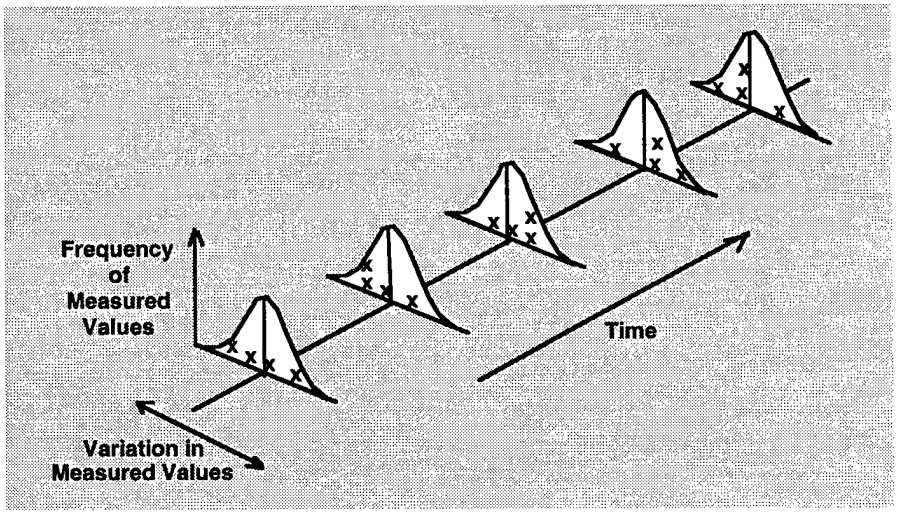


Figure 2-3: The Concept of Controlled Variation

The key word in the paragraph above is "predictable." Predictable is synonymous with "in control." Processes can vary in known, nonrandom ways and still satisfy Shewhart's definition of a controlled process. One example that Shewhart gives is that of the distance covered in successive intervals of time by a freely falling body [Shewhart 31]. Here our ability to predict the results is extremely precise, so by Shewhart's definition this is a

controlled process, even though the distance traveled increases steadily from interval to interval. The stock market, on the other hand, is not a controlled process.

The processes that we use in business and industry are rarely in control. As Shewhart points out,

> *...in the majority of cases there are unknown causes of variability in the quality of a product which do not belong to a constant system. This fact was discovered very early in the development of control methods, and these causes were called assignable.*
>
> *Walter A. Shewhart, 1931*

Variations due to assignable causes have marked impacts on product characteristics and other measures of process performance.[3] These impacts create significant changes in the patterns of variation. This is illustrated in Figure 2-4, which we have adapted from Wheeler and Chambers [Wheeler 92]. Assignable cause variations arise from events that are not part of the normal process. They represent sudden or persistent changes in things such as inputs to the process, the environment, the process steps themselves, or the way in which the process steps are executed. Examples of assignable causes of variation include shifts in the quality of raw materials, inadequately trained people, changes to work environments, tool failures, altered methods, failures to follow the process, and so forth.

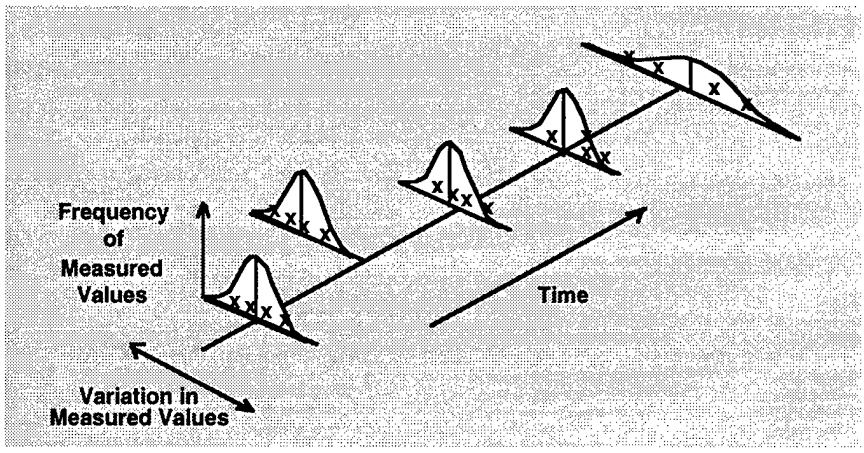


Figure 2-4: The Concept of Uncontrolled or Assignable Cause Variation

When all assignable causes have been removed and prevented from recurring so that only a single, constant system of chance causes remains, we have a stable process. The law of

[3]Assignable causes are sometimes called special causes, a term introduced by W. Edwards Deming.

large numbers then applies, and we can assume that the objective probability that such a cause system will produce a given event is independent of time [Shewhart 31].

Stability of a process with respect to any given attribute is determined by measuring the attribute and tracking the results over time. If one or more measurements fall outside the range of chance variation, or if systematic patterns are apparent, the process may not be stable. We must then look for the causes of deviation, and remove any that we find, if we want to achieve a stable and predictable state of operation.

One technique that is often used to establish operational limits for acceptable variation is statistical process control (SPC). SPC and its associated control charts were developed by Walter A. Shewhart in the 1920s to gain control of production costs and quality. Shewhart's techniques were used extensively in World War II and again in more recent years by W. Edwards Deming and others as a basis for improving product quality, both in Japan and in many U.S. companies. As a result of the successes of SPC in industrial settings, the techniques have been adopted for use in many other business areas.

To some, SPC is a way of thinking, with tools attached. Its aim is to improve processes through causal analysis. SPC differs from other, more traditional approaches—such as problem management, zero defects, and repairing and reworking products after they have been produced—in that it provides more constructive help in reducing the causes of defects and improving quality.

The principal thesis of this guidebook is that the techniques of SPC can be applied to software processes just as they have been to industrial processes, both to establish process stability and predictability and to serve as a basis for process improvement.

For example, Figure 2-5 shows a control chart for the number of reported but unresolved problems backlogged over the first 30 weeks of system testing. The chart indicates that the problem resolution process is stable, and that it is averaging about 20 backlogged problems (the center line, CL, equals 20.4), with an average change in backlog of 4.35 problems from week to week. The upper control limit (UCL) for backlogged problems is about 32, and the lower control limit (LCL) is about 8. If future backlogs were to exceed these limits or show other forms of nonrandom behavior, it would be likely that the process has become unstable. The causes should then be investigated. For instance, if the upper limit is exceeded at any point, this could be a signal that there are problems in the problem-resolution process. Perhaps a particularly thorny defect is consuming resources, causing problems to pile up. If so, corrective action must be taken if the process is to be returned to its original (characteristic) behavior.

We must be careful not to misinterpret the limits on the individual observations and moving ranges that are shown in the control chart. These limits are estimates for the *natural limits of the process*, based on the measurements that were made. The natural process limits together with the center lines are sometimes referred to as the "voice of the process."

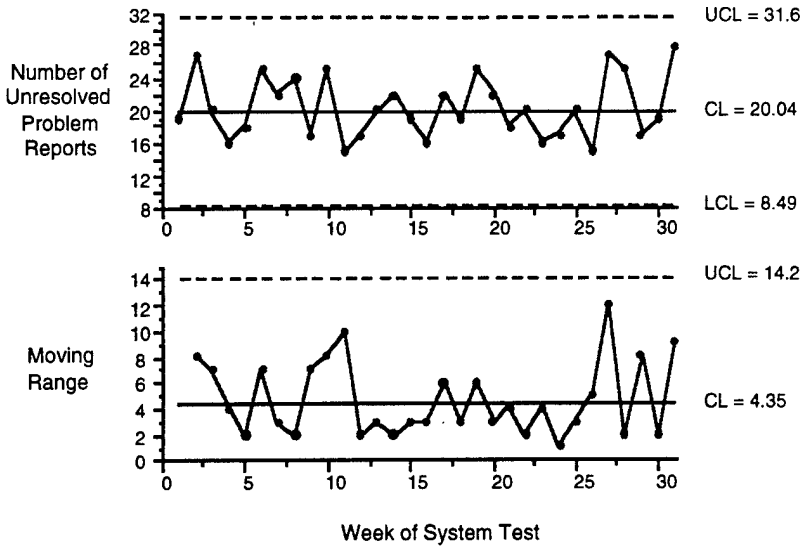


Figure 2-5: Control Chart for the Backlog of Unresolved Problems

The performance indicated by the voice of the process is not necessarily the performance that needs to be provided to meet the customer's requirements. If the variability and location of the measured results are such that the process, albeit stable, produces too many nonconforming products, the process must be improved. This means reducing the variability, moving the average, or both. We will illustrate these alternatives in Section 2.4 when we address the issue of process capability.

2.3 Compliance

For a software process to be stable and predictable, it must be operated consistently. Processes that are clearly defined, effectively supported, faithfully executed, reinforced, and maintained are more likely to be stable than those that are not. If a process is not stable, the search for assignable causes often begins with an examination of process compliance. There are three aspects of compliance that experience has shown can significantly affect process performance. Each of these aspects is measurable.

- **fitness (ability) to execute the process**. We measure fitness to allow us to understand the extent to which our organization is able and ready to use a particular process. If, for example, project personnel are not aware of, trained in, or given the tools needed for executing the process, it is unlikely that the process will be executed properly. Similarly, if the tools and procedures don't accomplish what was intended, the performance of the process will be in jeopardy. Understanding the fitness of the organization to execute the process allows us to identify appropriate actions to take to remedy situations where lack of fitness is the root cause of process instability or inability to meet customer needs.

- **use of the defined process**. Process stability depends on consistent execution of the process. If we determine that the performance of the process is not stable (not in-control), one reason may be that the process is not being executed as defined. If the intended tools are not being used, or if procedures other than those contained in the process definition are sometimes substituted, the process is likely to be unstable. By measuring the extent of defined process use, we can determine when the process is not being followed faithfully and what the reasons for deviations might be. We can then take appropriate action. Compliance with standards, directives, and other constraints may also concern us from time to time, but only if it bears directly on process performance.
- **oversight, benchmarking, and assessment of the process**. All processes are subject to forces of entropy; that is, if left alone, processes tend to deteriorate from a controlled state to a chaotic state. Unless we make a conscious effort to sustain our processes by reinforcing their proper use and by repairing the effects of entropy, we tend to loose control. Management oversight through frequent and periodic reviews of the process status, formal assessment of processes, and project benchmarking provides impetus for sustaining the process.

The concept of fitness to execute a process applies to people, resources, methods, technologies, and other support related to the process. Although an organization may have defined a process, there is no guarantee that its people can actually use it or properly execute it. There are many reasons why this may be so. Examples include lack of awareness, insufficient training, inadequate tools or facilities, difficulties in transferring technology, inadequate management support, and mismatches between the process and the software domain characteristics. Measurements help detect and diagnose these conditions.

Measuring the fidelity of process use involves measuring the breadth of use (how widely used), depth of use (thoroughness), frequency of use (how often used), and duration of use (how long in use). Evaluating process use requires that we have criteria in mind that must be met in order for a process to qualify as being "used." Asking questions such as, "Do you use tool Y?" or "Do you use procedure X?" is not very helpful in obtaining insights on use. Instead, questions that ask, "How often do you use X?", "How many use Y?", and "What and how much training has been received?" lead to descriptions and measurements of attributes that provide better insights.

Oversight, benchmarking, and assessment of a process are techniques that can be used to deal with process entropy. The measures used for this aspect of compliance will always evolve and vary significantly from one organization to another. The CMM, for example, defines five levels of process maturity that are used by many organizations to assess the maturity of their software processes [Paulk 93a, 93b]. Other models and measures can be found in the literature as well.

In practice, the attributes that organizations measure to determine compliance depend largely on the sensitivity of the results to different values of the attributes and on the likelihood that the attributes may vary. In many cases, a single variable or a small set of variables will dominate the others. Identifying the dominant variables is important because of the high degree of leverage, in terms of measurement cost, that they offer for recognizing assignable causes of instability and opportunities for process improvement.

Compliance measures are not measures of performance in that they are not attributes that reflect the degree to which the process fulfills its purpose. Instead, compliance measures address reasons why a process might not be performing as it should. These measures provide contextual information for interpreting and acting on the data used to characterize process performance. If used properly, compliance measures also provide early warnings of processes that are about to go out of control.

Figure 2-6 lists some software entities and attributes that are good starting points for measurement when examining questions of process fitness and use. The entities of interest

Fitness Are all the requisites for successful process execution in place? (personnel, skills, experience, training, facilities, tools, documented procedures, etc.)		**Use** Is the process being executed faithfully? Are the tools, methods, practices, and work plans being used?	
Entities	**Attributes**	**Entities**	**Attributes**
• People	Skills Experience Training Quantity	• People	Effort Time
• Tools	Accessibility Adequacy Utility Skills	• Tools	Use How widely How frequently How long
• Procedures	Coverage Sufficiency Quality Documentation	• Procedures	Awareness Use How widely How frequently How long
• Facilities	Space Computers Technical Support Sufficiency	• Facilities	Use How widely How frequently How long
• Work Plan	Targets Work Breakdown Structure Applicability Understandable Doable	• Work Plan	Awareness Use How widely How frequently How long

Figure 2-6: Examples of Entities and Attributes that Can Be Measured to Address Process Compliance

are people, tools, procedures, facilities, and work plans. The attributes of interest for each entity vary with the questions. Different questions often require different measurement approaches. When you use Figure 2-6 as a guide, you should add to or modify the entities and attributes so that they address the processes of concern in your own organization.

Figures 2-7 and 2-8 are examples of charts produced from compliance measures. The issue being examined is the penetration of inspection practices across all divisions of a major company. Here a survey was conducted to investigate the extent to which software and document inspections were being used [Grady 94]. The year was 1993; each project was produced in a software laboratory; and each laboratory was located within a division. Engineers were asked to report on only the most recent project they completed. A project was credited with using software inspections if it had held four or more inspections over the life of the project. The project was credited with using a particular type of document inspection if it used that type at least once.

The percent of projects that used four or more inspections was computed for each laboratory. Because there were reasons to suspect that results might vary with laboratory size, the results were grouped according to the number of software (SW) and firmware (FW) engineers in the laboratory, and the average percent of projects using inspections within each group was determined. The shaded bars in Figure 2-7 show the results. The unshaded bars show how the engineers were distributed across the company (e.g., about 50% of the company's engineers worked in laboratories that employed 75 or more engineers).

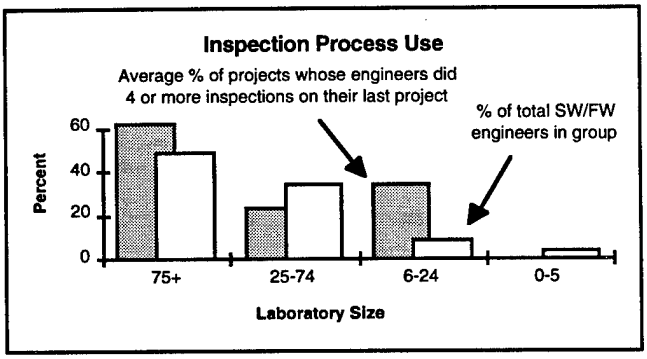


Figure 2-7: Inspection Process Use

Figure 2-8 shows the extent to which documents were inspected in those divisions that reported that more than 25% of their recent projects used software inspections four or more times (i.e., divisions that had been at least modestly penetrated by software inspection practices). The bars show the percent of projects in these divisions that reported using the indicated types of inspection one or more times.

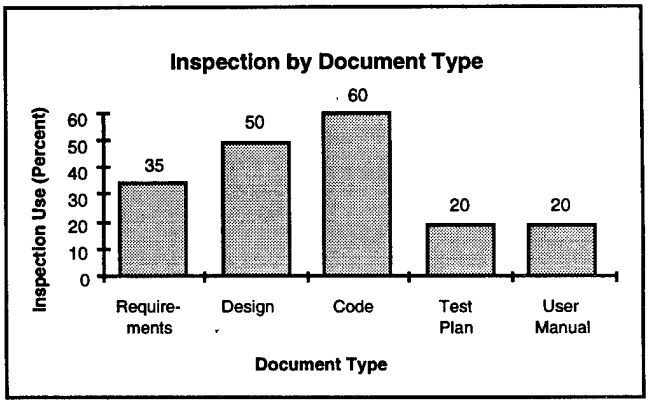


Figure 2-8: Inspection Use by Document Type

Measurements of compliance are not intended to address process performance. Their primary purpose is to provide contextual information that helps to explain performance results and the variations and patterns that are observed. For example, if a sequential plot of a product or process measure suggests that the process has become unstable, compliance measures may point to the cause. On the other hand, if a process is unstable while being executed in compliance with its definition, this may be a strong indication that the problems lie elsewhere.

2.4 Capability

> *A process has no measurable capability unless it is in statistical control.*
>
> *W. Edwards Deming, 1986*

What is process capability? As in the case of stability, we must recognize that measured values of most process and product characteristics will vary over time. When a process is stable (under statistical control), its results will have predictable means and be within predictable ranges about the means. We can then examine the question of whether or not the process is meeting its goals.

The histogram in Figure 2-9 is a frequency plot of measurements obtained from a stable process. The histogram shows the empirical distribution of measurements of elapsed time between a problem's evaluation and its resolution in a problem-resolution process. Since the process is known to be stable (presumably determined by control charting), the results can be taken to reflect the ability of the process to perform in a timely fashion. As the variation is

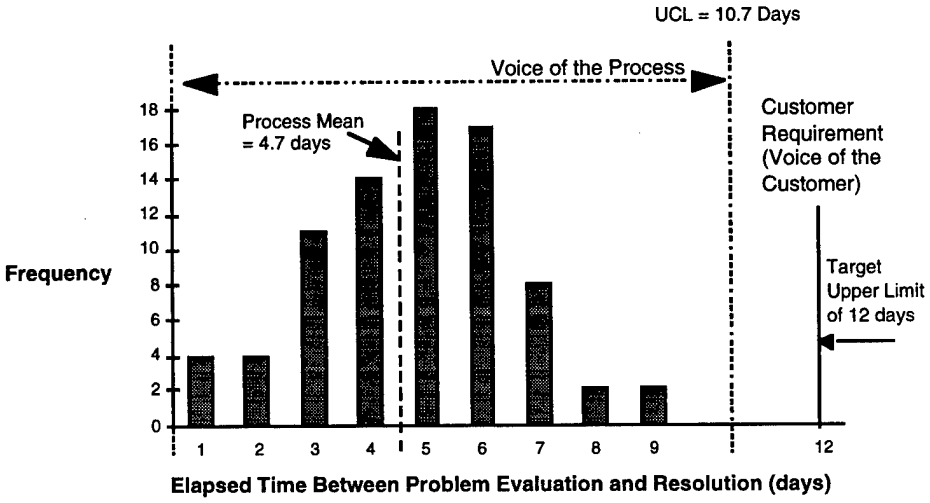


Figure 2-9: Histogram Reflecting Process Capability

due solely to a constant system of chance causes (the definition of stable), the mean and control limits shown in the figure represent what the process is able to accomplish, as it is currently defined and executed. In this sense the histogram, the mean, and the limits together represent the voice of the process.

If we superimpose the customer (or business) requirement on the histogram, we can compare what the process says it can do to what the business says it needs. For example, the solid vertical line to the right of the histogram in Figure 2-9 depicts the customer's requirement. Requirements (or specifications) like this are often termed "the voice of the customer."

If, on the other hand, one or both of the natural process limits were to fall outside the specification limits, there would be frequent occasions where the process would not meet its requirements. To change that situation, either the variability would have to be reduced or the process mean would have to be moved (or both). A third alternative may exist in some situations—relaxing the specifications to bring the requirement and the process limits into alignment. Figure 2-10 illustrates the three alternatives.

When the variability shown by the voice of the process falls within the limits required by the customer, the process *conforms* to customer requirements. When a process is stable and conforming to requirements, it is termed *capable*. The concept of capability thus depends on both the stability of the process and its ability to conform to customer requirements.

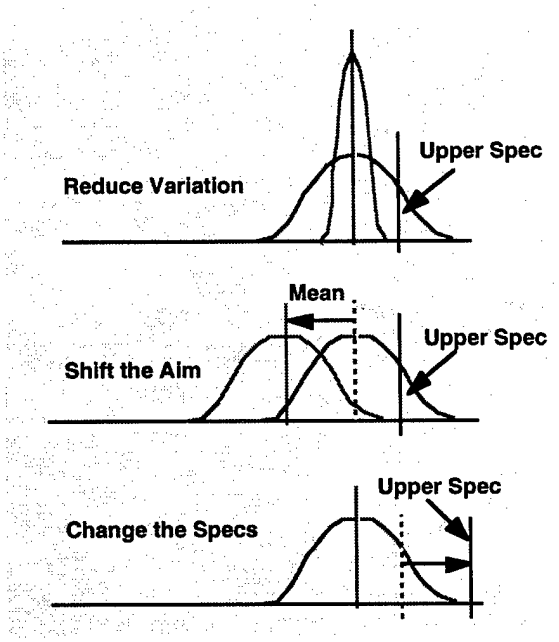


Figure 2-10: Aligning Process Performance to Process Requirements

Analysis of process performance identifies areas where the capability of a process can be improved to better support business objectives. Even though a process may be stable, its capability may need to be improved to satisfy competitive pressures or comply with customer needs.

Note that the concept of process capability is different from that of a product tolerance or planning target. Mistaking product tolerances and planning targets for process capability is a frequent cause of missed commitments.

Knowledge of process capability is essential for predicting the quality of products and the proportion of defective units that an organization produces. Projections of quality that are based on desires or wishful thinking, rather than statistically relevant historical data, are destined to cause problems down the road.

2.5 Improvement and Investment

Any company engaged in continuous improvement strategies will be able to substantiate its improvements by observing a large number of associated out-of-control processes.

W. Edwards Deming, as reported by Hoyer and Ellis, 1996

Business goals and strategies, together with factual data about attributes of product quality and process performance, are the key drivers that lead to actions that improve a software process. But just identifying the changes that are needed to improve a process is not sufficient. We must also look at the benefits that justify the costs associated with changing the process. This often means that measures which go beyond process performance may be required. Examples include training costs and costs for additional equipment, new programming tools, and so forth.

Economic payback may not be the only measure appropriate for quantifying the value of process changes. Recent studies have concluded that traditional financial indicators alone may be neither necessary nor sufficient to judge business performance or to justify investment in process changes [Kaplan 92, Brodman 95]. These studies cite operational measures of customer satisfaction, organizational innovation, competitive advantage, and improving core competence as important drivers of future business success.

For example, Lynch and Cross, in their book *Measure Up!* suggest a number of measures that go beyond traditional financial measures like profitability, cash flow, and return on investment [Lynch 95]. The measures that they propose relate to business operating systems, and they address the driving forces that guide the strategic objectives of the organization (see Figure 2-11).

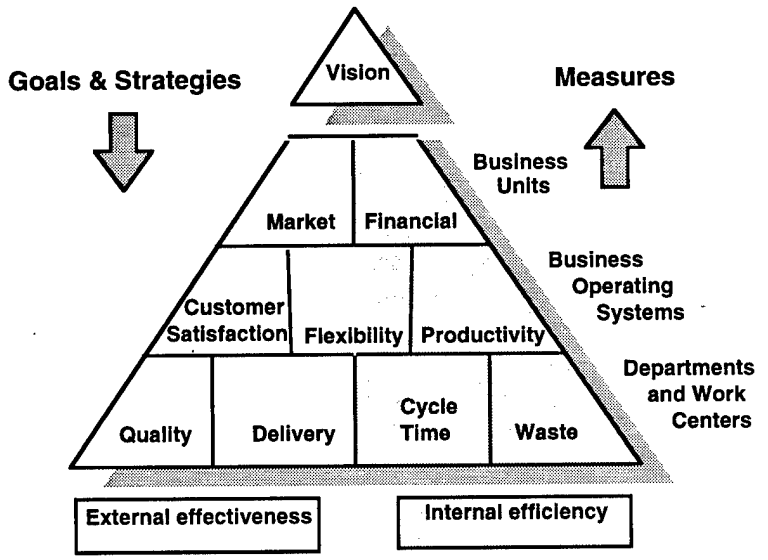


Figure 2-11: Lynch and Cross's Performance Pyramid for Business Operating Systems

Lynch and Cross suggest that customer satisfaction, flexibility, and productivity are the driving forces upon which company objectives are based. Status of these factors can be monitored by various indicators, as suggested in Figure 2-12, which is adapted from their

book. These indicators can be derived, in turn, from lower level (departmental) measures of waste, delivery, quality, and cycle time. The concept is illustrated in Figure 2-13. Figure 2-13 also points out that many measurements of process performance are obtained directly from measurements reported by the projects that make the organization's products.

The measures that you should use to rationalize and gain approval for software process investment and improvement are fundamentally the same as those suggested by Lynch and Cross. Departmental (and lower level) measures can be related to software product and process measures with little difficulty. Rework, defect counts, schedules, development time, and service response time are just a few examples.

Potential Indicators of Customer Satisfaction, Flexibility, and Productivity	
Customer Satisfaction	License renewal rate
	Number of new licenses
	Revenue per customer
	Number of new customers
	Number of complaints
	Customer ratings of products or services (from surveys)
Flexibility	Quoted lead times
	On-time delivery
	Time to market
	Time to accommodate design changes
	Number of change requests honored
	Number of common processes
	Number of new products
Productivity	Reductions in product development or service cost
	Rework as a percent of total work
	Cost-to-revenue ratios
	Ratios of development time to product life

Figure 2-12: Examples of Indicators for Business Operating Systems

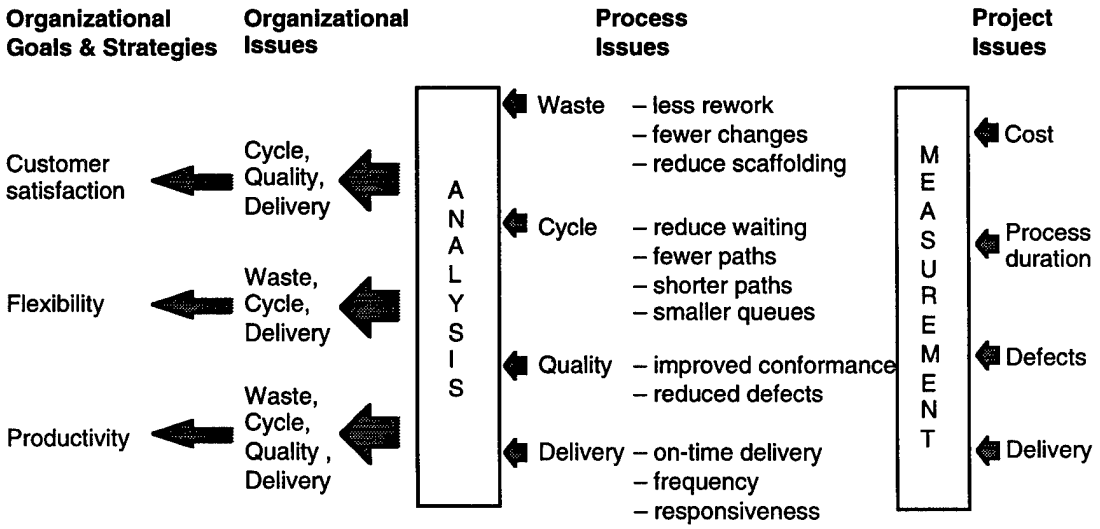


Figure 2-13: Sources of Early Indicators for Customer Satisfaction, Flexibility, and Productivity

It is important, therefore, that people who propose improvements understand not only the business goals and strategies of the organization, but also the priorities, risks, and issues associated with the goals and strategies. Identifying process measures that correspond to the business indicators in Figures 2-12 and 2-13 is challenging but doable. The key is to relate the costs and benefits of process improvement actions to the business indicators used in your own organization.

Experience has shown that measurements and analyses which can withstand the scrutiny of people not directly involved with the management and improvement of software processes are essential. One of the most important issues is that of demonstrating in a practical and statistical sense that the changes proposed can reasonably be expected to have the effects predicted. When we are able to show that process performance is stable and that changes are therefore needed if process capability is to be improved, and when we can predict the benefits of proposed process improvements based on sound evidence, then the credibility and confidence in proposals that require investment are enhanced significantly. These proposals then have a much better chance of being accepted and adequately funded. Proposals and plans that can be shown to measurably aid achievement of organizational goals are more readily approved and adapted than those that cannot.

3 Planning Measures for Process Management

The situation was a paradise for theorists, untrammeled by facts.

J. Donald Fernie, 1996

In Chapter 1, we outlined the responsibilities and objectives of process management and identified the relationships that the activities of *planning* and *applying* measures have with process management. The purpose of this chapter is to explore measurement planning activities in more detail and show how they relate to process management. This focus is highlighted in Figure 3-1.

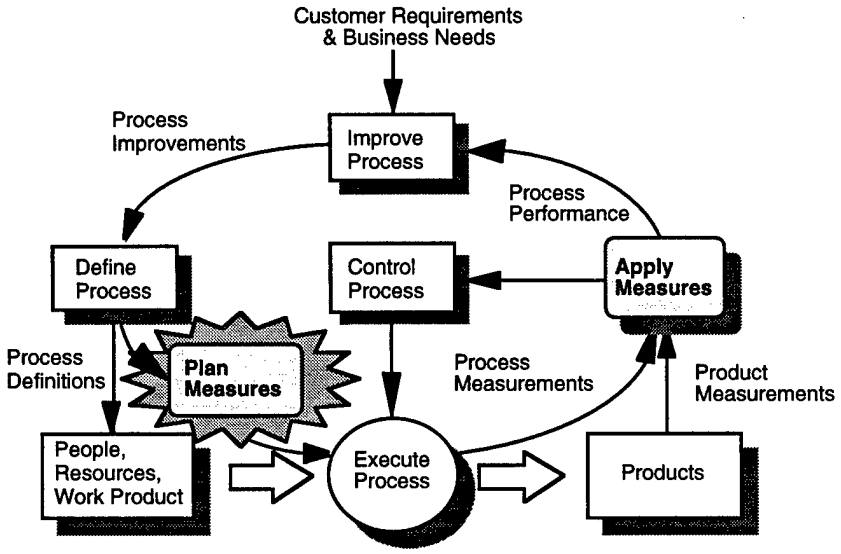


Figure 3-1: Measurement Planning and the Key Responsibilities of Process Management

Measurement planning progresses in three stages: *identifying* process management issues, *selecting and defining* the corresponding product and process measures, and *integrating* the resulting measurement activities into the organization's existing software processes. Figure 3-2 illustrates the sequence of these stages.

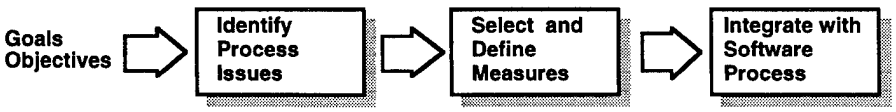


Figure 3-2: The Principal Activities of Measurement Planning

Each stage in Figure 3-2 builds on the one that precedes it. The goals and objectives of your organization will guide you in identifying issues; identifying your issues leads to selecting and defining measures; and selecting and defining measures puts you in position to plan the related measurement activities and integrate them into your software processes.

Sections 3.1–3.3 describe the three stages of measurement planning and suggest approaches to each.

3.1 Identifying Process Issues

> *All too often our perception is wrong and the answer to our question was OK, but we simply asked the wrong question.*
>
> *Jim Barr, 1996 (in an internet posting to comp.software-eng)*

Process management responsibilities can encompass the entire life cycle of a software product. They can also focus on specific aspects of the development or support cycle. In either case, the process or subprocess that is being managed has a purpose or objective—a *raison d'etre* that can be traced back to the software organization's business goals.

Experience has shown that it is important to identify the critical factors that determine whether or not your processes succeed in meeting the goals you set. These critical factors often arise from concerns, problems, or issues that represent levels of risk that threaten your ability to meet your goals, responsibilities, or commitments. They also arise from specifications that must be met to satisfy customer requirements or downstream process needs. We refer to critical factors collectively as issues. Note that issues are not necessarily problems. Rather, based on your understanding and experience with the processes and the products the processes produce, they describe situations that require attention.

Steps for Identifying Process Issues

The following steps outline a straightforward approach for identifying process issues.

1. **Clarify your business goals or objectives.** You will need to understand how your business goals or objectives relate to your software processes. In most cases, business goals and objectives that are tied to cost, quality, or time can be mapped readily to the appropriate software processes.
2. **Identify the critical processes.** Processes that have experienced problems in the past, are executed across organizational boundaries, use a technology for the first time, or operate under higher work loads than have been used before are all prime candidates for your list of critical processes. Processes that provide inputs or support to downstream processes are candidates as well. The list you identify may vary with the

passage of time or with your progress through a development or support cycle.

3. **List the objectives for each critical process.** Listing objectives in terms of process performance will help you identify potential problem areas. The best way to do this is in terms of the attributes of the products or processes that you want to control or improve. You may find that successfully meeting the objectives for a process depends on the performance of upstream processes. If these processes have not been selected for measurement, you may want to consider adding them to the list created in Step 2.

4. **List the potential problem areas associated with the processes.** These are concerns or issues that could jeopardize attaining the objectives. Listing potential problem areas requires that you have a good understanding of your processes and how they relate to each other. Process flow diagrams and listings of the entities associated with them can help significantly here.

5. **Group the list of potential problems into common areas or topics.** This will help you to identify issues that can be described and quantified by closely related measurements.

The Role of Mental Models

One technique for identifying key process issues is to ask questions about the performance of each process and the quality of the products it produces. Your ability to identify critical factors and issues depends on the picture you have in your mind of the tasks, resources, and products required to meet your goals. This picture is often called a *mental model* [Senge 94]. Sketching or diagramming your mental models for the processes you manage can be very helpful in communicating the issues and measurement possibilities to others, as well as in solidifying your own understanding of the processes. An explicit model of the process under study will visually summarize the relationships among the process tasks. This often brings to light implicit factors that are otherwise overlooked and that may need to be understood more clearly.

Figure 3-3 illustrates the general shape of many process models. Generic models like this are easily elaborated and adapted to specific situations. To construct a specific process model, you should look for

- things the process receives (inputs and resources)—these are used or supplied
- things the process produces (outputs)—these include products, by-products, and effects
- things the process consists of (activities, flow paths, and agents)—the structure of the process

- things that are expended or consumed (consumables)
- things the process holds or retains (internal artifacts, such as inventory and work in progress)

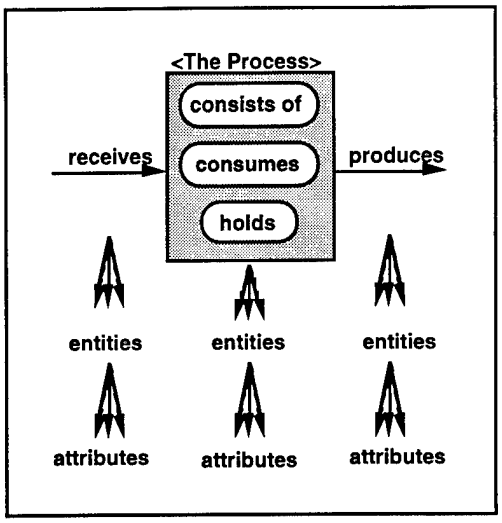


Figure 3-3: A Generic Process Model

Each "thing" in the process is an entity, and each entity has attributes that characterize some aspect of the process or its products. Thus, every attribute in a model of a process is a candidate for measurement. We will return to this generic model and discuss it in more detail later in this chapter.

Figure 3-4 shows a simple model for a problem management process. The lines from one box to the next indicate the flow of problem reports. The boxes designate the major tasks that must be carried out to move each problem report from the "open" state to the "closed" state. Simple models like Figure 3-4 are useful not only for identifying process issues that bear watching, but also for selecting process attributes for measurement. They are also useful for ensuring that everyone working on process improvement is addressing the same process.

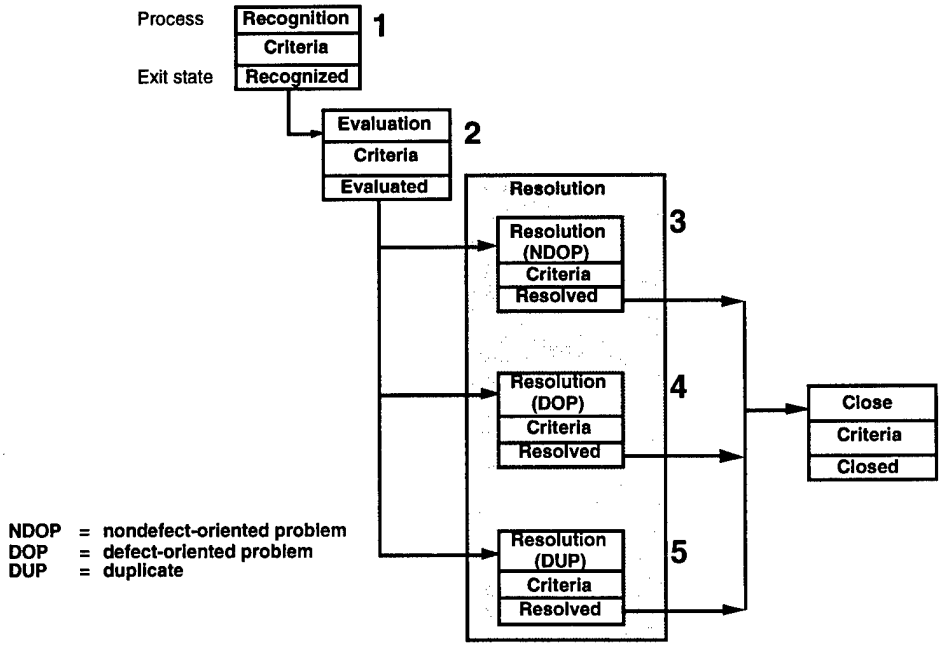


Figure 3-4: A Simple Process Model for Defect Tracking

Once you have sketched flowcharts or other pictures that describe a process, step back a bit and examine the process as a whole to see if you have missed anything. By asking questions such as the following, you may discover additional entities whose properties could be worth measuring.

- What determines product quality?
- What determines success?
- What do our customers want?
- What could go wrong?
- What's not working well?
- What might signal early warnings?
- Where are the delays?
- How big is our backlog?
- Where is backlog occurring?
- What things can we control?
- What limits our capability?
- What limits our performance?

As your knowledge of a process increases, or when the process is inherently large and complex, you may find more sophisticated methods useful. For example, structured control-flow diagrams like the one illustrated in Figure 3-5 can help you to identify important elements of a process and establish mutually understood frameworks for team-based efforts [Henry 92].

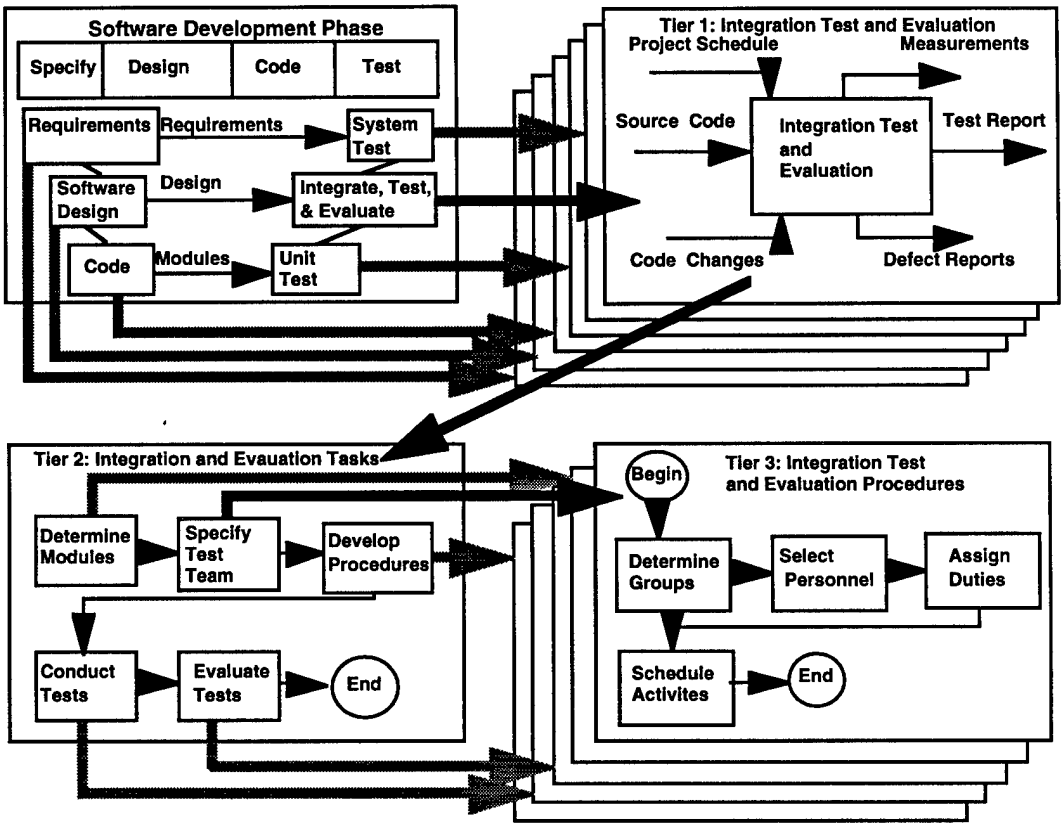


Figure 3-5: Example of a Control-Flow Diagram

Common Process Issues

All processes have at least three, and often four, characteristics in common. In addition to producing a specific product or service, they require expenditures of resources and time to produce the product or service; they are expected to deliver the product on time; and from time to time they will produce products with defects or imperfections. Because of these similarities, there are certain fundamental issues associated with processes that everyone concerned with process management shares. These are

- product quality (specifications, tolerances, action limits, and defects)
- process time or duration
- product delivery
- process cost

These issues tie very closely to the process performance attributes that an organization will want to select for control or improvement. The issues—important to most organizations and common to all software processes—are discussed briefly below. They are readily measured, and understanding the issues provides motivation for all participants in the process.

- **Product quality (specifications, tolerances, action limits, and defects).**[4] Excessive variability and off-target processes cause defects. Defects introduced into a product have multiple effects. They require the effort of skilled personnel to detect, remove, repair, and retest. Defects also increase process cost, time, and complexity. In addition, defects that escape detection and repair before the product is released to a customer reduce user satisfaction with the product. This is just as true for products that go to internal customers as it is for those that go to external customers. Activities that decrease the introduction of defects or increase the early detection of defects are prime targets for measuring the effectiveness of the process.
- **Process duration.** Process duration is an issue that translates directly to a process attribute. It is the elapsed time from the start of processing until the product is available to the user. The user in this case is not necessarily the customer or "end user." It may be a person or team involved in subsequent process activities. The flow of work through a given process can often be improved by providing better tools, using better technology, making more effective use of time, eliminating unnecessary steps, and improving training.
- **Product delivery.** Delivery is a process event. At delivery, the timeliness and quantity of a product or service can be measured against expectations

[4]Defects occur when products do not meet specifications for designated attributes of product quality. Tolerances are the amount of leeway granted by specifications. Action limits are limits used to control variability in specified attributes of product quality.

or commitments. If a product or service is not on time (late or early), this may have consequences for the customer or user of the product or service. Delivery is directly related to process duration (via lead time and the time required to execute the process) and delivery frequency.

- **Process cost**. Process cost includes the cost of executing and supporting the process as well as the costs of understanding and managing the factors that drive costs. As in any other activity, the cost of a software process has many elements and is affected by a variety of factors. Examples include the nature of the product, the statement of work, skills and experience of personnel, labor rates, cost of facilities, overhead, and the pacing of the process. The key to managing process cost is to identify the elements of cost that you can control or affect and look for opportunities to reduce or eliminate the costs. Because of the labor-intensive nature of producing and supporting software, effort is frequently the predominant cost driver for software processes. Measuring how and where people spend their time and relating this to causal factors is important in understanding and controlling the cost of a process [Perry 94]. This often leads to recognizing the need for better tools, training, and different skill mixes.

3.2 Selecting and Defining Measures

> *...there is nothing more important for transaction of business than use of operational definitions.*
>
> *W. Edwards Deming, 1986*

We now turn to the second planning activity, *selecting and defining measures*, as highlighted in Figure 3-6.

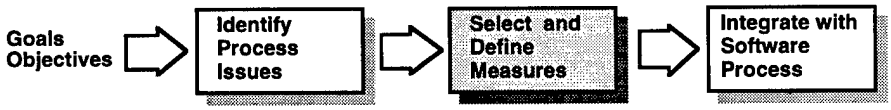


Figure 3-6: Measurement Planning Activities—Step 2

Selecting and defining measures are two different but closely related activities. As we will see, it is one thing to select a measure such as the number of reported defects that we might use to characterize a process or product. It is quite another to create an operational definition for a measure once it has been selected.[5] For example, an operational definition

[5]Operational definitions tell people *how* measurements are made, and they do this in sufficient detail that others will get the same results if they follow the same procedures.

for the number of defects found might include the type of defect, when it occurs, when it is detected, the finding activity, whether the defect is static or dynamic, its criticality, and a description of the processes used to find, classify, and count the occurrences of the defect. To have an operational definition, you must know enough about what the data represent and how they are collected (1) to ensure that different people will implement the measures correctly and consistently and (2) to interpret the data correctly.

The activities of selecting and defining measures are coupled in practice because it is not unusual to have to reconsider a measure due to your organization's inability to apply it easily or consistently once it has been operationally defined. In the pages that follow, we point out several factors to consider when selecting measures. Then we discuss some important characteristics of operational definitions and suggest methods for making your definitions operational.

As you select and define your measures, keep in mind that you will be seeking to apply them to help you control and improve your processes. The primary vehicles that will help you do this are statistical process control and quality improvement methodology. Nonmanufacturing applications of SPC and quality improvement often require ingenuity beyond that normally required for the manufacturing applications where the methods have demonstrated proven value. There seem to be two main reasons why this is so [Montgomery 96]:

1. Most nonmanufacturing operations do not have natural measurement systems that allow us to define quality easily.
2. The systems that are to be improved are usually fairly obvious in a manufacturing setting, while the observability of processes in a nonmanufacturing setting may be fairly low.

The key to process management and improvement in nonmanufacturing environments is to focus your initial efforts on resolving these two issues.

Selecting Process Measures

> *...if one is to make a formal measurement, one must accept some responsibility for making some effort to define one's purpose. As in many other types of activity, this step is too often taken for granted. The result is that someone else who wants to use the measurement may have to struggle to interpret the data gathered by the original investigation. It even happens that the researcher himself forgets and changes his viewpoint, so that he later makes use of data in a way that is difficult to justify.*
>
> *Paul Kirchner, 1959*

In Section 3.1, we stressed the importance of identifying key process issues and the things you would like to know to better manage those issues. You must now decide what measures to use to shed light on the issues. As you think about this, you will realize that identifying issues is like the "tip of the iceberg." Selecting measures requires more than simply stating an issue. For instance, you want to know

- What processes have effects on the issue?
- What products are affected by the issue?
- What product entities and attributes should be measured?
- What process entities and attributes should be measured?
- What resource entities and attributes should be measured?
- How will the measurements be used?
- How will the measurements be analyzed?
- Who will use the measurement results?

Selecting measures can be a hectic and time-consuming activity if you do not have a clear understanding of the factors that can influence your selection. The following procedure can help you develop an understanding of many of these factors, so that you can choose measures that shed light on the issues. Keep in mind that the goal is to find measures that provide information relevant to the issues you identified in the initial stage of your planning efforts.

1. **Clarify the issue**. Make sure that you understand all facets and dimensions of the issue. Put the issue in context by
   - listing the questions that are being asked
   - listing the questions that need to be answered (even if they are not asked)
   - identifying who is asking and why (to ensure that you understand the perspectives of those using the measurement results)
   - identifying the time elements relative to the issue (Is the issue periodic, transient, or event based?)
   - identifying the purpose of the measures relative to the issue (What does the issue suggest in terms of data analysis and action? Will the data be used to understand, compare, predict, control, assess, or improve some aspect of the process?)
2. **Identify the processes encompassed by the issue**. You have done some of this already when you identified the critical process issues that concern you. You may now see other processes whose results affect the issue.
3. **Review the relationships among the processes** to determine which processes need to be measured. Some processes, although encompassed by the issue, may contribute little or no risk or uncertainty relative to the issue. Those processes can often be ignored. Sketches that make explicit

the mental models for your processes will be of significant help, both here and in the steps that follow.

4. **For each process, select the entities and attributes to be measured and the data elements to be collected.** That is, select attributes whose measures will be most influential or dominant in determining product quality or in meeting process objectives. Your knowledge of the process will be a significant factor in selecting entities and attributes that are important. You should try to identify entities and attributes that have direct relationships to process results. If you are interested in process cost, product size, number of defects found, and so forth, measure these attributes directly. Do not rely on indirect relationships to measure process results. For example, do not measure defects to determine complexity, or size to determine cost, unless you have a theory or data to support the validity of a particular cause-effect or predictive relationship.

5. **Test the potential usefulness of the selected measures** by sketching indicators, especially charts and graphs, that show how you propose to use the measurements you will obtain. Do not be surprised if this leads you to sharper, more focused questions and definitions.[6]

When selecting entities, attributes, and data elements for measurement, it is helpful to frame your selections in the context of a model that describes the process of interest. Visual models are especially helpful. With a concrete (explicit) model of the process as your guide, you can solidify your understanding of the process as well as communicate with others when selecting entities to measure. Good models will help you identify the products and by-products that are produced at all stages of a process.

Some examples of the kinds of entities often found in processes are listed in Figure 3-7. You may want to include some of these elements in your mental models and consider measuring one or more of them when seeking to understand what your processes are doing and why they are doing it.

We suggest that you tuck Figure 3-7 away for future use as a checklist. It gives an extended (but by no means complete) set of entities to consider when looking for useful product and process measures. Note that characterizations of process performance will usually be based on attributes of entities found in the four right-most columns. Measures of process compliance, on the other hand, will usually address entities found in the two left-most columns. (Entities in the second column can be of interest in both cases.)

Figure 3-8 is similar to Figure 3-7. It gives examples of attributes of process entities that you may want to consider as candidates for measurement when studying process performance.

---

[6]Examples of indicators and their relationships to measurement definitions and mental models, are given in *Goal-Driven Software Measurement—A Guidebook* [Park 96a].

Measurable Entities in a Software Process

Things received or used	Activities and their elements	Things consumed	Things held or retained	Things produced
products and by-products from other processes	processes and controllers	resources	people	products
ideas, concepts	• requirements analysis	• effort	facilities	• requirements
resources	• designing	• raw materials	tools	• specifications
• people	• coding	• energy	materials	• designs
• facilities	• testing	• money	work in process	• units
• tools	• configuration control	• time	data	• modules
• raw materials	• change control		knowledge	• test cases
• energy	• problem management		experience	• test results
• money	• reviewing			• tested components
• time	• inspecting			• documentation
guidelines and directions	• integrating			• defects
• policies	flow paths			• defect reports
• procedures	• product paths			• change requests
• goals	• resource paths			• data
• constraints	• data paths			• acquired materials
• rules	• control paths			• other artifacts
• laws	buffers and dampers			by-products
• regulations	• queues			• knowledge
• training	• stacks			• experience
• instructions	• bins			• skills
				• process improvements
				• data
				• good will
				• satisfied customers

Figure 3-7: Measurable Entities in a Software Process

Neither Figure 3-7 nor 3-8 should be viewed as a formal taxonomy or classification scheme. Their purpose is simply to guide you as you look for elements and attributes that can provide useful information for managing or improving process performance. In practice, it makes little difference whether or not you have things classified "right" (whatever that means). But it makes a big difference if you have overlooked something important. Reviewing Figures 3-7 and 3-8 may give you some ideas and help avoid oversights.

When using Figures 3-7 and 3-8 as checklists, keep in mind that inputs to one process are almost always outputs of another—things received or used have to be produced somewhere, and things produced presumably get received and used by some downstream customer or activity. Depending on the process, activity, or task that you are examining, you may find elements in column 1 that belong in column 5 and vice versa. If so, simply put them where you think they belong. The important thing is not where they fit, but that you found something worth measuring.

Measurable Attributes of Software Process Entities

Things received or used

changes
- type
- date
- size
- # received

requirements
- requirements stability
- # identified
- % traced to design
- % traced to code

problem reports
- type
- date
- size
- origin
- severity
- # received

funds
- money
- budget
- status

people
- years of experience
- type of education
- % trained in XYZ
- employment codes

facilities & environment
- square feet per employee
- noise level
- lighting
- # of staff in cubicles
- # of staff sharing an office or cubicle
- investment in tools per employee
- hours of computer usage
- % of capacity utilized

Activities and their elements

flow paths
- processing time
- throughput rates
- diversions
- delays
- backlogs

length, size
- queues
- buffers
- stacks

Things consumed

effort
- # of development hours
- # of rework hours
- # of support hours
- # of preparation hours
- # of meeting hours

time
- start time or date
- ending time or date
- duration of process or task
- wait time

money
- cost to date
- cost variance
- cost of rework

Things produced

status of work units
- # designed
- # coded
- # tested

size of work units
- # of requirements
- # of function points
- # of lines of code
- # of modules
- # of objects
- # of bytes in database

output quantity
- # of action items
- # of approvals
- # of defects found

test results
- # test cases passed
- % test coverage

program architecture
- fan in
- fan out

changes
- type
- date
- size
- effort expended

problems & defects
- # of reports
- defect density
- type
- origin
- distribution by type
- distribution by origin
- # open
- # closed

critical resource utilization
- % memory utilized
- % cpu capacity utilized
- % I/O capacity utilized

Figure 3-8: Measurable Attributes Associated with Software Process Entities

It is important to note that many measurements that can be used to quantify process performance are the same as those used for project management. The attributes in Figure 3-8, for example, are often important not just to process management, but to program and project management as well. The document *Practical Software Measurement: A Guide to*

Objective Program Oversight [JLC 96] contains similar lists of things to measure. Many of the measures in that document can be used for managing and improving processes too.

In the final analysis, selecting process measures comes down to understanding the purpose and operation of the process and determining the inherent issues (potential problems and risks) that may prevent the process from meeting its purpose. In most cases, these issues will have an underlying basis in one or more of the common issues described in the previous section. In these situations, we are frequently led to selecting certain core measures—such as defect counts, cost, and time—to quantify the issues.

Recall that our discussions in Chapter 2 pointed out that measures of product and process quality can be used to achieve process stability and determine process capability. The measures that you select to quantify management issues are often exactly the ones that you use to determine if a process is stable and capable. Remember that when you introduce changes to improve a process, you will need to measure the results to verify that process performance or capability has changed or improved.

As you consider which entities and attributes to measure, you should begin to formulate goals for the measurement activities that will have to be implemented to get the information you need. Well-structured measurement goals define the following elements [Rombach 89, Shepperd 93]:

- **the object of interest** (an entity). The object of interest may be a product, process, resource, agent, artifact, activity, or environment. It may also be a set or collection of other entities or objects. In short, any "thing," real or abstract, that you want to describe or know more about is a potential object for measurement.
- **the purpose**. The purpose of a measurement activity may be to understand, predict, plan, control, compare, assess, or improve some productivity or quality aspect of the object.
- **the perspective**. The perspective identifies who is interested in the measurement results. It identifies a role such as developer, maintainer, manager, or customer. The perspective is stated to clarify the purpose of the measurement activity.
- **the environment and constraints**. A description of the environment provides a context for interpreting measurement definitions and results. When the context is not made explicit, it may not be understood by all who collect or use the reported data. The chances for misusing the data are then substantial, as it is very easy to implicitly assume things that are not so and thus arrive at erroneous conclusions.

Templates, examples, and a process for identifying and stating measurement goals are illustrated in *Goal-Driven Software Measurement—A Guidebook* [Park 96a].

Defining Process Measures[7]

Data without definitions are indistinguishable from numbers.

– source unknown

Once you have identified your measures, you must define them: names alone will not suffice. You must be able to tell others exactly how each measure is obtained so that they can collect and interpret the values correctly.

Measurement of software products and processes is not new. Some organizations have been measuring for years. At a minimum, we have all dealt with schedules. Many organizations have also recorded effort expenditures, perhaps weekly, if for no other reason than to ensure that employees get paid. Some organizations use these data in conjunction with measures of software artifacts to track and control progress, especially when developing products under contract. Some of these organizations have structured estimating processes that use empirical models to help them translate records from past projects into bids, proposals, and plans for future work.

But despite all this measurement activity, few in the software industry would call measurement a success story. This is especially true when we attempt to use data that were collected or reported by someone else. Some reasons for our lack of success are

- **Different users of measurement data have different needs**. Data collected for one purpose may not be suitable for another, because the rules used for collecting the data are inconsistent with the ways others want to use the data.
- **Different organizations have different established practices**. In many cases these practices have sound reasons behind them and should not be changed. Moreover, it may be difficult and often impractical to change the way an organization collects data, just to satisfy an external need.
- **Unambiguous communication of measurement results is inherently difficult**. Even if someone understands perfectly well how their data are collected, it is not easy for them to communicate adequate descriptions of the operational rules to others. These rules may be complex, and they may never have been stated explicitly.
- **Structured methods for communicating measurement results seldom exist**. What you think you hear is often not what they meant to say. This, in a way, restates the ambiguity point just made, but frames it in a way that suggests a potential solution.

---

[7]The discussions here are taken from *Goal-Driven Software Measurement—A Guidebook* [Park 96a]. Further information, together with guidelines for leading teams through the process of identifying, defining, and implementing measures, are illustrated in that guidebook.

Our proposal, then, is to use checklist-based frameworks to help define, implement, and communicate operational definitions for software measures. The primary issue is not whether a definition for a measure is correct, but that everyone understands, completely, what the measured values represent. Only then can we expect people to collect values consistently and have others interpret and apply the results to reach valid conclusions.

Communicating clear and unambiguous definitions is not easy. Having structured methods for identifying all the rules that are used to make and record measurements can be very helpful in ensuring that important information does not go unmentioned. When designing methods for defining measures, you should keep in mind that things that do not matter to one user are often important to another. This means that measurement definitions—and structures for recording the definitions—often become larger and more encompassing than the definitions most organizations have traditionally used. This is all the more reason to have a well-organized approach. Definition deals with details, and structured methods help ensure that all details get identified, addressed, and recorded. They also help you deal with people who believe that attention to detail is no longer their responsibility.

Criteria for Operational Definitions

> *An operational definition puts communicable meaning into a concept.*
>
> *W. Edwards Deming, 1986*

Operational definitions must satisfy two important criteria [Park 92]:

- **communication.** If someone uses the definition as a basis for measuring or describing a measurement result, will others know precisely what has been measured, how it was measured, and what has been included and excluded?
- **repeatability.** Could others, armed with the definition, repeat the measurements and get essentially the same results?

These criteria are closely related. In fact, if you can't communicate *exactly* what was done to collect a set of data, you are in no position to tell someone else how to do it. Far too many organizations propose measurement definitions without first determining what users of the data will need to know about the measured values in order to use them intelligently. It is no surprise, then, that measurements are often collected inconsistently and at odds with users' needs. When it comes to implementation, rules such as, "Count all noncomment, nonblank source statements" and "Count all open problems" are open to far too many interpretations to provide repeatable results.

Although communicating measurement definitions in clear, unambiguous terms requires effort, there is good news as well. When someone can describe exactly what has been

collected, it is easy to turn the process around and say, "Please do that again." Moreover, you can give the description to someone else and say, "Please use this as your definition, but with these changes." In short, when we can communicate clearly what we *have* measured, we have little trouble creating repeatable rules for collecting future data.

Examples of Operational Definitions

> *Making intuition more explicit opens it to scrutiny, repetition, refinement. Good things!*
>
> *Stan Rifkin, 1997 (in an internet posting to comp.software-eng)*

Frameworks for constructing operational definitions for some frequently used size, effort, schedule, and quality measures have been described in three SEI technical reports [Park 92, Goethert 92, Florac 92]. The frameworks are based on checklists, supplemented by forms for summarizing operational information that is not amenable to checklist treatment.

An example of a checklist that has been used to define the counting of problems and defects found during system testing is shown in Figure 3-9. Here the checkmarks in the *Include* and *Exclude* columns spell out the rules to be followed when deciding whether or not a particular problem, defect, or report is to be included in a given count. Similarly, once a count has been made, the checkmarks tell you exactly what has been counted and what has not. The *Value Count* and *Array Count* columns of the checklist provide structures for requesting specialized counts, either for subsets of the total or for tables (arrays) that cross tabulate the frequencies of occurrence for sets of results from different attributes.

The central theme in the checklists lies in stating exactly what is included in—and excluded from—reported results. The checklists are supported by supplemental forms that describe how the inclusions and exclusions were (or are to be) accomplished. These practices should be part of an operational definition, since they affect the way measured results should be interpreted.

Although the first (and most important) use of definition checklists and supplemental forms is to let users of data know exactly how the data were obtained, the same kinds of descriptions can be used to specify how future measurements are to be made. The latter "let me tell you what to do" approach is the one we usually see in software organizations, but without visible structures for ensuring that the measurement instructions will be interpreted and executed consistently by all who collect the data.

Formats for checklists like those in Figure 3-9 should be tailored to the particular problem-tracking process that is used within your organization. Since these processes and the terms that they employ vary from organization to organization, you should make sure that the checklists you use to define problem and defect counting fit your needs. This is true for your other measures as well.

Problem Status	**Include**	**Exclude**	**Value Count**	**Array Count**
Open	✔		✔	
Recognized				✔
Evaluated				✔
Resolved				✔
Closed	✔		✔	
Problem Type	**Include**	**Exclude**	**Value Count**	**Array Count**
Software defect				
Requirements defect	✔		✔	
Design defect	✔		✔	
Code defect	✔		✔	
Operational document defect	✔		✔	
Test case defect		✔		
Other work product defect		✔		
Other problems				
Hardware problem		✔		
Operating system problem		✔		
User mistake		✔		
Operations mistake		✔		
New requirement/enhancement		✔		
Undetermined				
Not repeatable/Cause unknown		✔		
Value not identified		✔		
Uniqueness	**Include**	**Exclude**	**Value Count**	**Array Count**
Original	✔			
Duplicate		✔	✔	
Value not identified		✔		
Criticality	**Include**	**Exclude**	**Value Count**	**Array Count**
1st level (most critical)	✔			✔
2nd level	✔			✔
3rd level	✔			✔
4th level	✔			✔
5th level	✔			✔
Value not identified		✔		
Urgency	**Include**	**Exclude**	**Value Count**	**Array Count**
1st (most urgent)	✔			
2nd	✔			
3rd	✔			
4th	✔			
Value not identified		✔		

Figure 3-9: A Checklist-Based Definition for Counting Defects (page 1 of 2)

Finding Activity	Include	Exclude	Value Count	Array Count
Synthesis of				
Design		✔		
Code		✔		
Test procedure		✔		
User publications		✔		
Inspections of				
Requirements		✔		
Preliminary design		✔		
Detailed design		✔		
Code		✔		
Operational documentation		✔		
Test procedures		✔		
Formal reviews of				
Plans		✔		
Requirements		✔		
Preliminary design		✔		
Critical design		✔		
Test readiness		✔		
Formal qualification		✔		
Testing				
Planning		✔		
Module (CSU)		✔		
Component (CSC)		✔		
Configuration item (CSCI)		✔		
Integrate and test		✔		
Independent V & V		✔		
System	✔			
Test and evaluate		✔		
Acceptance		✔		
Customer support				
Production/deployment		✔		
Installation		✔		
Operation		✔		
Undetermined				
Value not identified		✔		
Finding Mode	**Include**	**Exclude**	**Value Count**	**Array Count**
Static (non-operational)	✔			
Dynamic (operational)	✔			
Value not identified		✔		

Figure 3-9: A Checklist-Based Definition for Counting Defects–continued (page 2 of 2)

Creating Your Own Definition Frameworks

There are of course many measures for which checklists and descriptive forms do not yet exist. When you propose measures that have no current checklists, you should develop similar (or equivalent) vehicles for communicating the rules and procedures that you want used to capture and record your data. Checklists are useful, especially when inclusion and exclusion decisions affect results.

Whatever frameworks you choose, your structured methods must tell the people who collect the data exactly what is to be included in (and excluded from) the values they report to you. Where it makes a difference—and it usually does—they must also describe how the measurements will be carried out. An appropriate definition framework ensures that any variation in the method for measuring that could affect either the values themselves or the way they should be interpreted gets described.

When constructing checklists and supporting forms for defining software measures, you will find that the best way to ensure full coverage and achieve consensus is to focus not on telling people what they should do, but rather on identifying what you and others need to know to use the data correctly. Not only will this minimize controversy and confrontation, but once you have a structure that communicates all relevant information about a measurement's result, it is easy to use that structure to tell others how to collect the data you want.

3.3 Integrating Measures with the Software Process

The third stage in measurement planning (highlighted in Figure 3-10) is to integrate your defined measures with your software processes. In this section, we discuss the three steps that comprise this stage: *analysis, diagnosis,* and *action. Analysis* involves identifying the measures that your organization collects now and how they are being used. *Diagnosis* means evaluating the extent to which these measures can be used to meet your newly identified needs and determining where additional work is needed. *Action* is directed at translating the results of *analysis* and *diagnosis* into action plans for collecting and using the additional data you seek. These steps are described in the paragraphs that follow.

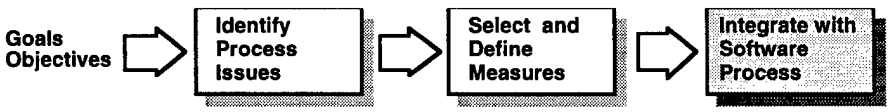


Figure 3-10: Measurement Planning Activities—Step 3

Analysis of Existing Measurement Activities

Analysis establishes the baseline for the work that follows. Knowing what measures your organization collects now and how people are using them gives you a starting point for implementing the measures you have defined. If your organization is like most, you will not

be starting from scratch. Some measurement activities will already be in place, and you should use these as springboards if you can. Revolutionary approaches often meet strong resistance. Where possible, it makes sense to build on things that are currently in use, strengthening them in the process, and refocusing them where necessary.

When analyzing your existing measures and measurement practices, you should ask questions such as

- What data elements are required for my measures?
- Which ones are collected now?
- How are they collected?
- Which processes provide the data?
- How are the data elements stored and reported?

Often you will find that there are more potential data sources than were apparent at first glance. The mental models that you create for your processes can help you locate these sources. For example, several sources often exist for data about defects and problems. The situation shown in Figure 3-11 is typical of many organizations [Florac 92]. Here people who build products (product synthesis) write problem reports; teams that inspect products (as in peer reviews) prepare inspection reports; participants in formal milestone reviews produce action items; test groups produce test reports; and customer support groups document customer problems. All of these reports are followed by analyses and corrective actions, and the results and status are usually recorded somewhere, often in one or more databases. You should find and examine these databases to see what they can give you.

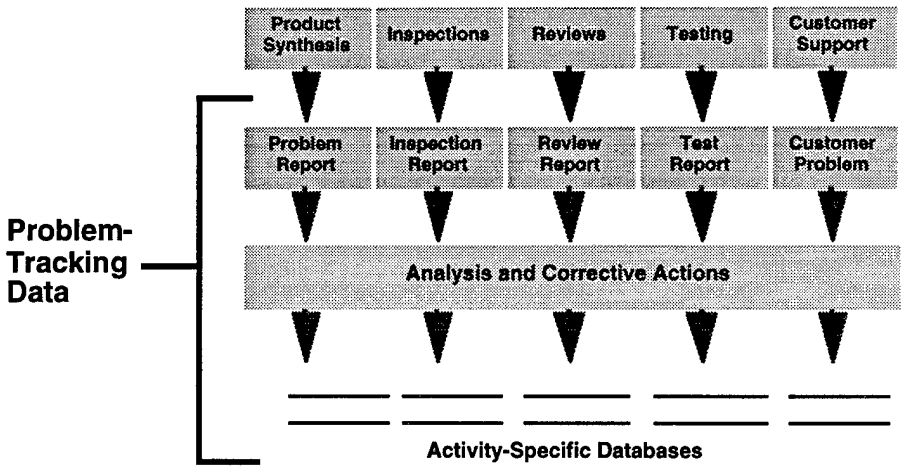


Figure 3-11: Sources for Problem-Tracking Data

Diagnosis of Existing Measures

Diagnosis means evaluating the data elements that your organization is collecting now, determining how well they meet the needs of your goal-driven measures, and proposing appropriate actions for

- using the data
- adapting the data to your needs
- adapting your needs to the data
- obtaining what is missing

Where analysis is fact-finding, diagnosis is evaluative and judgmental. When diagnosing, you are identifying alternatives and setting the stage for finding solutions. You are asking questions such as

- What existing measures and processes can be used to satisfy our data requirements?
- What elements of our measurement definitions or practices must be changed or modified?
- What new or additional processes are needed?

Action to Integrate Measurements

Action means translating the results of your analyses and diagnoses into implementable steps. It is concerned with finding solutions and making the solutions happen, and it includes identifying tasks, assigning responsibilities and resources, and following through to make sure that the actions happen.

Action starts with identifying the elements that you will build on or address in your measurement plans. Some things you will want to do before writing your plans are

- Identify the sources of data within your existing software process(es).
- Define the methods that will be used to collect and report the data.
- Identify (and specify) the tools that will be required to collect, report, and store the data.
- Determine your requirements for points in time and frequencies of measurement.
- Document your data collection procedures in detail:
  - Identify responsible persons and organizations.
  - Determine where, how, and when to collect and report.
  - Create sketches for the data collection records you will use.
- Determine who will use the data.

- Define how the data will be analyzed and reported.
- Prepare a data definition and collection process guide.

You should also analyze your data storage and access requirements. This includes identifying or determining

- your historical retention needs
- who will collect, store, maintain, and access the data
- the organizational levels to be served (Serving more than one organizational level often translates into a need for more than one database.)
- the granularity of the data (Will the data be retained as initially recorded or aggregated in some way?)
- the procedures to be used for dynamically editing and verifying data as the data elements are entered into the database
- the number of people with access to the data
- the need for recording the definitions associated with the data, so that users can tie the data to the descriptive information that is needed to use the data correctly

In addition, you should pay close attention to issues of data privacy wherever you encounter them. This is especially important for data that could be used (or perceived to be used) to evaluate the performance of individuals or teams. Much anecdotal evidence exists to suggest that the surest way to make measurement fail is to have people suspect that the measures might be used against them.

Tasks for Defining Your Measurement Processes

When you are preparing to write your measurement plans, it helps to have a checklist such as the one in Figure 3-12 to ensure that nothing gets overlooked. This checklist can be transformed easily into a display for summarizing your status with respect to defining the measurement process you intend to implement, as illustrated in Figure 3-13. Here codes have been used to show the status of each task. Actions to complete the unfinished tasks are things that you will want to address as you prepare your plan.

Checklist for Preparing a Measurement Action Plan

- Define the data elements.
- Define the scales to be used for measuring and recording observed values for each data element.
- Define the frequencies of collection and the points in the process where measurements will be made.
- Define the timelines required for moving measurement results from the points of collection to databases or users.
- Create forms and procedures for collecting and recording the data.
- Define how the data are to be stored and how they will be accessed. Identify who is responsible for designing the database and for entering, retaining, and overseeing the data.
- Determine who will collect and access the data. Assign responsibilities for these actions.
- Define how the data will be analyzed and reported.
- Identify the supporting tools that must be developed or acquired to help you automate and administer the process.
- Prepare a process guide for collecting the data.

Figure 3-12: Checklist for Preparing a Measurement Action Plan

Action Plans

Once you know what you have to start with (analysis), how well your present measures meet your business needs (diagnosis), and the actions that you will to take to meet the remaining needs (action), you are ready to prepare plans for implementing the actions you have identified. Your next step is to write the plans. These plans should translate your objectives into operational actions. They should make the measurement objectives clear; describe the goals and scope of the efforts and their relationships to other functional activities; spell out the tasks, responsibilities, and resources needed; provide for progress tracking and risk management; and establish frameworks and resources for sustained successful operation and evolution.

A template for assembling measurement action plans is presented in Appendix B. Keep in mind that there is no need to wrap all your measurement activities into a single, monolithic plan. You are not trying to establish an empire—you just want to get and use the data you need. Several small plans may be easier to implement and sustain than a single all-encompassing endeavor. As time passes, your needs will change. With a modular planning strategy, you will find it easier to update, augment, or replace your plans when the need arises.

Planning tasks	Data element 1	2	3	4	5	6	7
Data elements defined	Y	N	60%	Not Doc'd	Y?	•	•
Data collection frequencies and points in the software process defined	50%	N	60%	Not Doc'd	•	•	•
Timelines defined for getting measurement results to databases and users	N	N	30%	Not Doc'd	•	•	•
Data collection forms defined	N	N	N	N	•		
Data collection procedures defined	N	N	•	•			
Data storage, database design, and data retention responsibilities defined	N	N	•	•			
Who will collect and who will access the data identified	N	N	•	•			
Analysis processes defined	N	N					
Reporting processes defined	N	N					
Supporting tools identified and made available	N	N					
Process guide for data definition and collection prepared	Y						

Figure 3-13: Status of Action-Planning Activities

4 Applying Measures to Process Management—Part 1: Collecting and Retaining Data

Important questions in science and industry are how and under what conditions observations may contribute to a rational decision to change or not to change a process to accomplish improvements. A record of observations must accordingly contain all the information that anyone might need in order to make his own decision.

W. Edwards Deming, 1986b

Of what value is the theory of control if the observed data going into that theory are bad?

Walter A. Shewhart, 1931

In the previous chapter, we discussed the measurement planning activities associated with process management. In this chapter and the three that follow, we turn our attention to *applying* the measures in ways that support the decisions associated with process management. Figure 4-1 highlights the focus of our discussions.

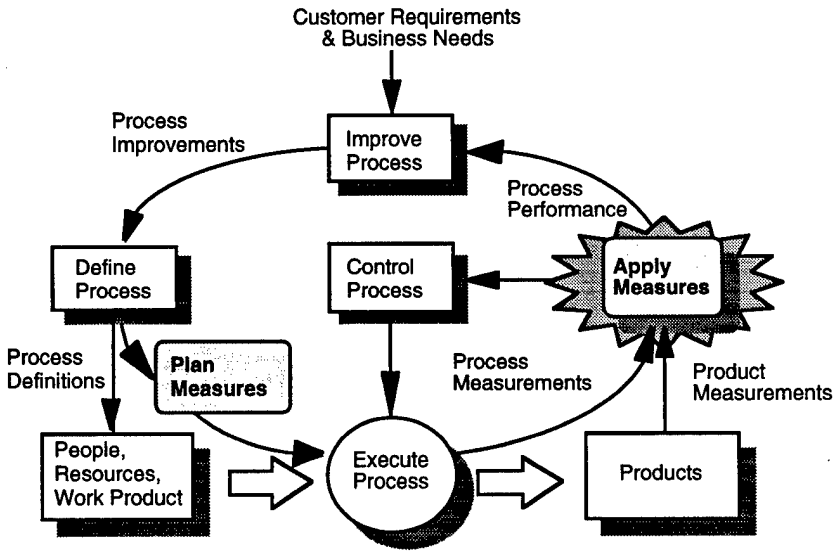


Figure 4-1: How Applying Measures Relates to the Key Responsibilities of Process Management

Applying measures is the operational phase of the measurement process. It consists of *collecting and retaining* process management data, *analyzing* the data, and *acting* on the results. Collecting and retaining data are prerequisites for analysis. Analyzing the data in the context of the issues at hand then leads to using the results to control and improve the software process and its subprocesses.

This chapter and the two that follow address the subactivities of applying measures in the sequence shown in Figure 4-2.

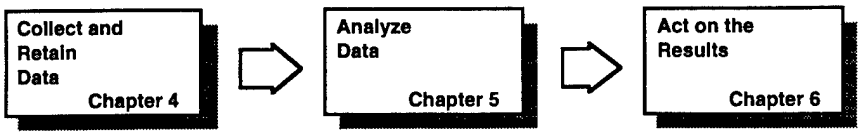


Figure 4-2: Applying Measures: The Subactivities

4.1 General Principles

> *An element of chance enters into every measurement; hence every set of measurements is inherently a sample of certain more or less unknown conditions. Even in the few instances where we believe that the objective reality under measurement is a constant, the measurements of this constant are influenced by chance or unknown causes. Hence, the set of measurements of any quantity, even though the quantity itself be a constant, is a sample of a possible infinite set of measurements which we might make of this same quantity under essentially the same conditions.*
>
> *From this viewpoint, measurement is a sampling process designed to tell us something about the universe in which we live that will enable us to predict the future in terms of the past through the establishment of principles or natural laws.*
>
> *Walter A. Shewhart, 1931*

The operational activities of measurement begin with collecting and retaining data. The procedures that you defined for collecting and retaining data must now be integrated into your software processes and made operational. This means putting the right people, sensors, tools, and practices into the processes in the right places. It also means capturing and storing the data for subsequent use in analysis and process improvement.

The principal tasks associated with collecting and retaining data for process management are as follows:

- Design the methods and obtain the tools that will be used to support data collection and retention.
- Obtain and train the staff that will execute the data collection procedures.
- Capture and record the data for each process that is targeted for measurement.
- Use defined forms and formats to supply the collected data to the individuals and groups who perform analyses.
- Monitor the execution (compliance) and performance of the activities for collecting and retaining data.

The following sections discuss these tasks in more detail.

4.2 Collecting Data

> *Getting management's help can be difficult. We had one manager who said we can gather any metrics we want as long as it doesn't cost him any money or bother his people. How's that for enlightenment?*
>
> *Dave Kleba, Abbott Laboratories, 1996*

Once you have selected and defined your measures and planned your implementation actions, you are ready to begin collecting data. Collecting data is more than just making measurements. It consists of implementing your plans, ensuring that they work, and sustaining the measurement activities that result. These actions will be facilitated if you document in detail your procedures for collecting, recording, and reporting data. Documenting your procedures involves

- identifying the responsible persons and organizations
- specifying where, when, and how measurements will be made
- defining the procedures to be used for recording and reporting results
- providing standard "fill-in-the-blank" forms to simplify manual recording of the data

The complexity of your data collection processes will increase as additional organizations or software processes become involved. Each organization or process may use a different tool or method to obtain the "same" data. If you wish to compare or aggregate the data you collect, you must ensure that the methods different people use to collect the data are, in fact, collecting exactly the same kinds of data. That is, your different tools and procedures should

be counting, extracting, or otherwise processing data in ways that produce equivalent results. The concept of *well-defined* data that was defined in Chapter 2 comes into play here. When it is possible to do so, providing standard tools such as code counters and work breakdown structures will make achieving consistency and common understandings easier.

Having a data collection guide that fully describes your data definitions and collection processes will shorten the amount of time required to stabilize your measurement results. It will also improve the quality of the data you collect.

Collecting data is a process. Like any other process, it must be monitored to ensure not only that the data are being collected, but that they are timely, complete, ungarbled, authentic, accurate, and otherwise of good quality. In short, if the results are to be reliable, the collecting process must be stable and under control. This implies that the performance of the measurement process itself should also be measured.

Process Performance Data

> *No count or measurement has any meaning apart from its context. The context for any value will always determine how that value should be analyzed and how it should be interpreted.*
>
> *Donald J. Wheeler, 1995*

> *The order in which the measurements were made should always be preserved in recording data for a frequency distribution.*
>
> *Grant and Leavenworth, 1996*

The fact that data will subsequently be analyzed can impose requirements on data collection in ways that may not be obvious. For example, measures of process performance that are used to assess process stability and capability require that special attention be paid to four important issues:

- **time sequence.** The order in which observations are made contains crucial information for estimating the inherent variability in a process. Moreover, knowledge of the sequence and of its relationships to time and process-related events is what enables you to identify the point where assignable causes of variation entered a process. This helps greatly in identifying the causes and preventing recurrences. You should ensure that any measurements collected for estimating, controlling, or improving process performance are accompanied by records of the sequence of observations. Where feasible, it helps also to relate the sequence to time and to any events or milestones that might affect measured values.

- **context data.** Analyzing control charts requires information about the context in which the data were produced in order to properly interpret the record of performance that is plotted on the charts. Thus, as pointed out in Chapter 2, process performance data should always be accompanied by information that permits questions like the following to be answered [Wheeler 92]:
  - What do the individual values represent? What are these numbers?
  - How were the values obtained? Who obtained them? When or how often?
  - What sources of variation are present in the data?

  Your data collection process must include practices that ensure that context data are captured when reporting product and process measurements.
- **rounding of data values.** Your collection processes must ensure that the scales for measuring and recording data are of appropriate granularity and that recorded values are not rounded inappropriately. Either condition can cause control charts to generate out-of-control signals even when the process is in a state of statistical control. A brief illustration of the types of problems that insufficiently precise measurement scales can cause is provided in Section 7.5 of this guidebook. Additional examples and guidelines related to this subject can be found under the topic of "Inadequate Measurement Units" in Wheeler's books [Wheeler 89, 92, 95].
- **measurement stability.** When you institute new measures or revise existing measures, your data collection procedures may have to be changed as well. This can result in destabilizing the measurement process. Testing and evaluating new or changed data collection processes with pilot runs can help shake out problems with the procedures and avoid collecting and retaining inappropriate data.

Process Compliance Data

Collecting compliance data typically requires seeking out sources other than those available directly from measures of operating processes. Two ways to obtain process compliance data are enumerated below. As always, the types and kinds of data required, together with the cost and availability of data, will determine which methods are best for you.

1. One way to investigate the extent of compliance is to review the process(es) in question. Generally this means conducting a series of structured interviews with project personnel. The responses can be combined with reviews of other organizational data such as budgets, reports, policies, product characteristics, or process documentation to provide measures of both fitness and use. This approach is particularly useful when looking for potential causes of instability or excessive

variability, as we pointed out in Chapter 2. It is also useful when establishing benchmarks for process improvement.

2. A second approach is to conduct periodic surveys where software managers and technical staff members answer questions regarding their compliance to the processes of interest. This approach is generally less expensive than the first approach, but the results may be less reliable due to differing interpretations and perceptions and to instincts for self-preservation among the people answering the survey. A well-designed survey can compensate for the inherent subjectivity in responses by asking questions that elicit related evidentiary data.

Process managers will often find it useful to conduct reviews or surveys to obtain a fitness profile of their project's personnel, tools, or methodologies. This can be especially true at the start of a project (or at periodic intervals in a long-term project) to help determine the training, tools, and technologies that may be needed to execute at the desired levels of process performance.

The use of surveys can be even more helpful at the organizational level. A survey is a relatively inexpensive and rapid method for gathering data to identify trends across projects. With careful survey design and by capturing relevant project, process, and environmental data and retaining the results over several survey periods, you can obtain trend information that can serve as a basis for stabilizing and sustaining process activities and for developing better tools and training. In addition, survey data can provide benchmarks and context data for project estimating and process improvement.

As just one example of the possibilities, Dillman describes an approach for self-administered surveys called the Total Design Method (TDM) [Dillman 83]. His experience suggests that response rates greater than 75% can be obtained with surveys averaging about 10 pages.

At both project and functional levels, capturing process compliance data dynamically can alert managers to deteriorating compliance and help head off process instabilities and performance problems before they occur.

4.3 Retaining Data

> *Data is like garbage. You better be sure what you're going to do with it before you collect it.*
>
> *Mark Twain (Apocryphal?)*

Retaining data inherently involves creating and using one or more databases to organize and save the data for later use. Depending on the nature of your measurement activities, this may be a reasonably simple task or a very complex and technically demanding one. In either case, it is important to give serious consideration to the data retention system that will

be employed. For example, while hard-copy forms may suffice for some data collection purposes, experience has shown that paper forms are often inadequate for retaining and aggregating measured results.

A personal computer database system and a full-functioned spreadsheet program may be sufficient for retaining and analyzing data for many processes. However, the size and complexity of the retention system will increase significantly if there is a need to support multiple projects or multiple organizations, or if you are using the measurement results for multiple purposes. The length of time you must retain the data can also influence your choice of a database system. For example, data for process management will often be retained and used well beyond the duration of individual projects.

A project management database, if it exists, may well serve as the basis for retaining process measurements for process management. This is something to be considered seriously before undertaking to develop a separate process management database, as there are often many overlaps between data collected for managing projects and data collected for managing processes.

You should consider the issues listed below when planning a process management database.

Database Planning Issues

Measurement Definitions

- The desire to standardize measures for data retention and analysis may conflict with software process tailoring or with the legitimate needs of projects to define and collect data in ways that address issues that are important to them. It may be unwise or impossible (or require added effort) to insist that all projects use the same measurement definitions.
- One alternative to standardizing measurement definitions is to permit freedom of definition (perhaps within prescribed limits), but to require standardized reporting via standardized formats for the definitions of the measures and measurement processes used.

Multiple Databases

- Do differing user needs, responsibilities, or levels of management require separate databases? If the answer is yes, a monolithic "software process database" is an unlikely choice, and the following issues arise:
  - How many databases?
  - Who will operate them, and where?
  - How will the databases be coordinated? (This involves addressing issues of concurrency, consistency, and propagation of data corrections and updates.)

Database Design Goals (Recommendations)

- Capture and retain definitions and context descriptions, not just direct measurement data.
- Tie measured values to measurement definitions, rules, practices, and tools used.
- Tie measured values to the entities and attributes measured.
- Tie measured values to the contexts and environments in which they were collected (product, environment, and process descriptors; process and project status; time and place measured; method of measurement; and so forth).
- Accommodate process tailoring (by recording descriptions of process specializations, tailorings, and other differences among processes).
- Accommodate evolving measurement definitions and process descriptions.
- Address linking to, accessing, and coordinating with other databases, such as those used for time and cost reporting, cost estimating, configuration management, quality assurance, personnel, and so forth.
- Avoid storing indirect measures (such as defect densities and rates of change) that can be computed by users from directly measured results. There are three reasons for this advice:
  - Storing computed values introduces redundancies in databases that are difficult to keep synchronized. When the underlying data change, indirect measures may not get recomputed.
  - If only the results of computations are stored, essential information easily becomes lost.
  - Other people may want to compute alternative indirect measures or compute them differently, so the underlying values will need to be retained anyway.

Logistical and Timeline Issues

- What are the media and mechanisms for moving data from the point of measurement to the database?
- How fast is the process from measurement to data entry? Will the data be timely and up to date?
- What are the provisions for coordinating the database with automated measurement tools? Can these provisions be automated?

Rules and Policy Issues

- What are your privacy objectives?
- What are your proprietary data objectives?
- What are your data access objectives?
- What are your retention objectives?
- What are your archiving objectives?

Database Operation Issues

Once you have settled the database planning issues, you should document your operational procedures in detail. This includes identifying

- who will enter and maintain the data
- who can access the data
- levels of access—for example, you may not want certain financial data to be available to everyone who has access to staff-hour time records.
- where the data will be retained
- the tools you will use, including the editing and retrieval mechanisms

Database Management Issues

Some additional database management issues that you should address are listed below. None of these issues are unique to software process management. We provide the list here to serve as a checklist and a reminder that the methods you use to retain and access data play a significant role in the success of any measurement activity.

Operating the Database

- responsibilities
- practices & tools for preventing simultaneous editing
- practices & tools for preventing contamination & corruption of previously verified data
- tools & training to support browsing, searching, retrieval, and display
- backup practices (frequency & off-site storage)
- funding
- training

Access

- Who is permitted to enter or change data?
- Who is permitted to access data?
- Who grants authority to access or change data?
- Who enforces access and change authority?
- What tools & practices will be used to support controlled access?

Entering Data

- responsibilities
- coordinating data entry with data verification
- funding
- training

Retaining Data

- where?
- how long?
- format & media
- backup practices (frequency and off-site storage)
- Who is responsible?
- funding

Archiving

- where?
- how long?
- format & media
- backup practices (frequency & off-site storage)
- Who is responsible?
- funding

Privacy (e.g., personnel data & personal performance data)

- Is there a need to isolate protected data from public data?
- Who grants authority to access protected data?
- Who enforces access rules?
- What tools & practices will be used to protect privacy?

Protecting Proprietary Data

- Who designates proprietary data?
- How is proprietary data identified?
- What tools & practices are used to protect proprietary information?
- Who can authorize access to or use of proprietary data?
- What are the ground rules for authorizing access?

Security (provisions for handling classified information)

- Is there a need for security procedures?
- Is multilevel security a requirement?
- What tools & practices will be used to protect security?

System Design

- hardware selection
- software selection
- database design (structure)
- communications among databases
- maintenance
- evolution
- operational support
- funding
- training

5 Applying Measures to Process Management—Part 2: Analyzing Data

...probability theory does not apply simply because a phenomenon is attributable to chance causes.

Walter A. Shewhart, 1931

Classical theory is based upon the concept of inference from a single sample from a statistical universe, the ordering within the sample being ignored, while control theory must be based upon evidence provided by a succession of samples, ordering within the sample, and other pertinent information.

Walter A. Shewhart, 1943

Analysis is the second major step in the operational phase of a measurement process. This chapter describes and illustrates some important analytic tools, and it shows how the tools can be used to help assess the stability, capability, and performance of a software process or activity. Figure 5-1 highlights the focus of the discussions.

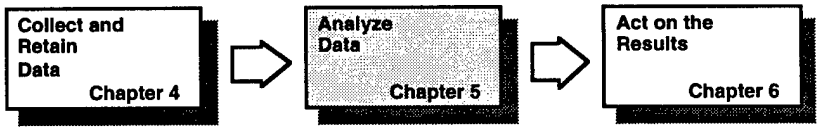


Figure 5-1: Chapter Focus : Analyzing the Data You Collect

The concepts and methods in the following sections deal primarily with the use of control charts for managing and improving process performance. Control charts are techniques for analyzing process stability, capability, and performance. They have been used with much success in other industries, but do not seem to have been widely adopted in software organizations. To be effective, control charts and the associated methods of statistical quality control should be used within the broader context of the goals you have established and the activities you perform to achieve those goals. The preceding chapters have set this context. The focus of this chapter is on the quantitative issues that underlie effective use of control charts for guiding process management and improvement actions.

5.1 Separating Signals from Noise

"...the method of attack is to establish limits of variability ...such that when [a value] is found outside of these limits, looking for an assignable cause is worthwhile."

Walter A. Shewhart, 1931

Data are generally collected as bases for action. No matter what the data or how the values are presented, you must always use some method of analysis to extract and interpret the information that lies in the data. Making sense of data is a process in itself. This is illustrated schematically in Figure 5-2.

Since data generally consist of both noise and signal, the values that are reported must be filtered somehow to separate the signals from the noise that accompanies them. This filtration may be based (subjectively) upon a person's experience and his or her presuppositions and assumptions, or it may be based on a more formalized approach. Without formal and standardized approaches for analyzing data, you may have difficulty interpreting and using your measurement results.

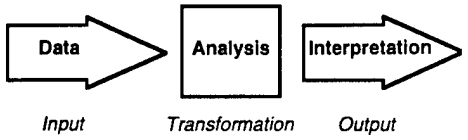


Figure 5-2: Interpretation Requires Analysis

When you interpret and act on measurement results, you are presuming that the measurements represent reality. Unless legitimate signals can be distinguished from the noise that accompanies them, the actions that you take may be totally unwarranted. Acting on noise as if it were signal serves only to amplify instabilities and increase the variability in process results. To use data safely, you must have simple and effective methods not only for detecting signals that are surrounded by noise, but also for recognizing and dealing with normal process variations when no signals are present.

The reason for analyzing process data is to draw inferences that can be used to guide decisions and actions. Drawing inferences (i.e., conclusions and predictions) from data depends not only on using appropriate analytical methods and tools, but also on understanding the underlying nature of the data and the appropriateness of assumptions about the conditions and environments in which the data were obtained.

There are two ways to obtain statistical inferences—from enumerative studies and from analytic studies [Deming 75]. The differences between the two are important and affect the kinds of inferences that can be drawn from measurement results. Techniques and methods of inference that are applicable to enumerative studies lead to faulty design and inference for analytic problems. The differences between enumerative and analytic studies are discussed in some detail in Section 7.1. The implications of the differences deserve considerable reflection before you begin interpreting and acting on the results of any analysis. We will defer discussing these implications, though, until you have a good understanding of some of the tools that are available to help you. For now, we simply point out that most analyses of process issues are analytic studies, and the methods in this chapter are aimed at guiding valid interpretations in these kinds of studies.

5.2 Evaluating Process Stability

> *Students are not warned in classes nor in the books that for analytic purposes (such as to improve a process), distributions and calculations of mean, mode, standard deviation, chi-square, t-test, etc. serve no useful purpose for improvement of a process unless the data were produced in a state of statistical control. The first step in the examination of data is accordingly to question the state of statistical control that produced the data. The easiest way to examine data is to plot points in order of production to learn whether any use can be made of the distribution formed by the data.*
>
> *W. Edwards Deming, 1986*

In Chapter 2, we introduced the concepts of process stability and process capability. We then illustrated the concepts in the context of measuring process performance. We return now to these topics and look at some of the methods that can be used to quantify and evaluate process performance.

The Importance of Stability

> *When a control chart indicates no special cause present, the process is said to be in statistical control, or stable. The average and limits of variation are predictable with a high degree of belief, over the immediate future. Quality and quantity are predictable. Costs are predictable. "Just in time" begins to take on meaning.*
>
> *W. Edwards Deming, 1993*

> *...if a control chart shows a process is "in control," it means that the hypothesis of random variation is a reasonable one to adopt for managerial purposes. When the chart fails to show control, then other action is reasonable.*
>
> *Acheson J. Duncan, 1974*

As we pointed out earlier, process stability is central to any organization's ability to produce products according to plan. It is also central to improving processes and to producing better and more competitive products. The facts are these:

- Without stability and the associated knowledge of what a process can do, we have no way to distinguish signals in measured values from the random

noise that accompanies them. Measurements then easily lead to inappropriate actions.

- Without a history of stable performance, uncontrolled excursions can occur at any time. We then have no rational basis for extrapolating observed performance to future situations, and all plans that we make are at risk.
- Without knowing what the stable level of performance is, we have no basis for recognizing assignable causes that signal improvement opportunities.
- Without stability, we have no repeatable process to use as a baseline for process improvement. In fact, some would say that we have not one process, but many—all different. Effects of improvement actions may then be indistinguishable from other assignable causes.

Therefore, to use product and process measures for predicting future results or as bases for process improvement (two instances of analytic studies), we must first ensure that the processes are behaving stably.

Stability Concepts and Principles

> *Original data plotted in the order of production may provide much more information than is contained in the distribution.*
>
> *W. Edwards Deming, 1986b*

In a stable process, the sources of variability are due solely to common causes." All variations in a stable process are caused by inherent factors that are part of the process itself. Variations due to "assignable causes," such as those caused by operator errors, environmental changes, deviations from the process, and changing characteristics in raw materials and resources, have been either removed from the process and prevented from reentering (if detrimental) or incorporated as a permanent part of the process (if beneficial).

A stable process is one that is in statistical control—the underlying distributions of its measurable characteristics are consistent over time, and the results are predictable within limits. This is illustrated in Figure 5-3. Each X in the figure represents a measured value of a given characteristic of a process or product. In this example, random samples of four units of a product were selected at five points in time. The locations of the Xs represent the actual measured values. Different values were observed at each sampling point due to inherent process variability (common cause variation). The curves in Figure 5-3 represent the distribution of the underlying variability. When a process is stable, the same curve represents the distribution of the product characteristic at each point it is measured. The process will remain stable as long as the distribution remains unchanged and unshifted.

To test a process for stability, we need to know how variable the values are within the samples at each measurement point, and we need to compare this to the variability that we

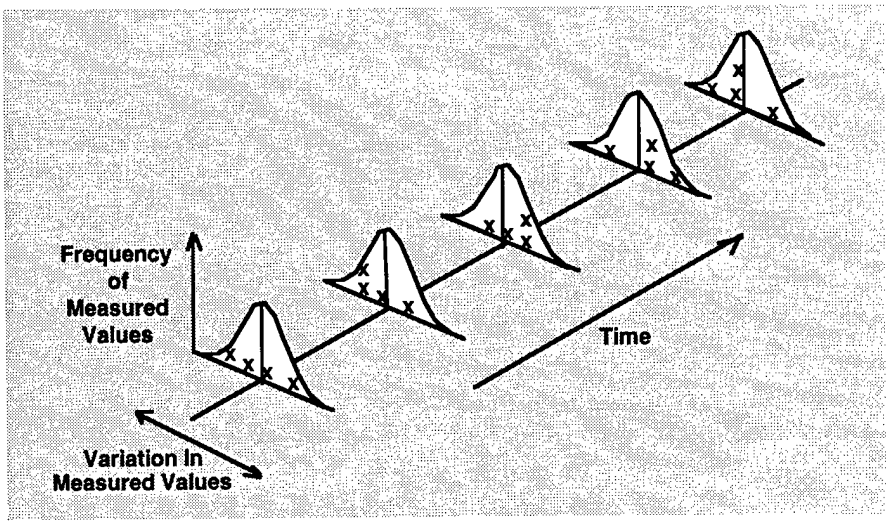


Figure 5-3: An Idealized Process in Statistical Control

observe from one sample to another. More specifically, we need to determine if the variation over time is consistent with the variation that occurs within the samples. We also need to detect possible drifts or shifts in the central tendency of the measured values. Shewhart's control charts provide a simple and effective means for conducting these tests. They are the statistical tools of choice to use when determining whether or not a process is stable [Wheeler 92, 93, 95].

Let's look again at Figure 5-3. If we have reason to believe that the variability at each sampling point is likely to be due solely to common cause variation, it makes sense to treat each sample as a homogeneous subgroup. We can then evaluate changes in central tendency by computing the *average* of the four measurements within each subgroup and plotting the results on a time scale. This lets us view the performance of the process in a time (or sequence) dimension.

The upper portion of Figure 5-4 shows the results for the values illustrated in Figure 5-3. This is called an X-bar or averages chart. The changes in the averages show the observed variation in central tendency from one subgroup to the next. Similarly, the *range* of the values within each subgroup can be calculated and plotted, as shown in the lower portion of Figure 5-4. This

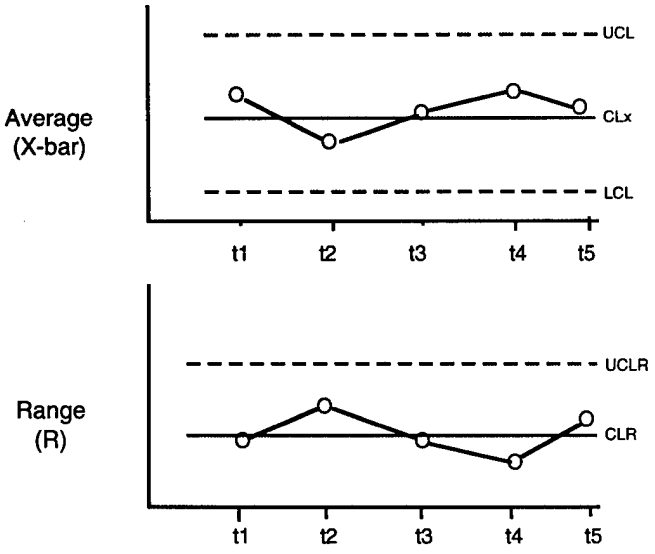


Figure 5-4: X-Bar and Range Charts for a Process That Is in Control

is called an R chart. R charts show the observed dispersion in process performance across subgroups.[8] Control limits for X-bar and R charts can then be calculated from the range data and subgroup averages. Calculating control limits is illustrated later in this chapter.

One of the most important principles (and skills) associated with control charting involves using knowledge of the process to select subgroups that contain, insofar as possible, only common cause variation. This is called rational subgrouping. The purpose of rational subgrouping is to permit the computation of estimates for standard deviations (values for sigma[9]) that are uncontaminated by assignable cause variations. This both narrows the control limits and provides the highest possible reliability for detecting unusual variations (signals) with a minimum of false alarms.

With control charts like the ones in Figure 5-4, we have a picture that tells us how the process has behaved with respect to a particular product or process characteristic over the period that was examined. The R chart tells us that the range of each subgroup (the difference between the smallest and the largest value in that subgroup) did not exceed the limits labeled UCLR (upper control limit, range). At the same time, the X-bar chart shows that none of the subgroup averages fell outside the upper or lower control limits (UCL and LCL) for averages of size 4. Finally, neither chart suggests any systematic or nonrandom patterns in the measured values over the period studied.

In short, plotting this set of data on X-bar and R control charts gives no "out-of-control" indications. This should be no surprise, since we stipulated at the outset that the process was in control. By using control charts, we have merely illustrated graphically the empirical conditions that must be satisfied for a process to be considered stable.

Figure 5-5, on the other hand, shows a process that is out of control. By plotting the measured values on averages and range (X-bar and R) control charts as in Figure 5-6, we see that the X-bar chart shows the average of the second subgroup to be above the upper control limit. This by itself is sufficient to suggest that the process is not in control. The subgroup range also exceeds its upper control limit at t5. This is another indication that one or more assignable causes may have affected the process, and that the process is out of control.

---

[8]Another form of chart called an S chart can also be used for this purpose, but the computations are somewhat more involved.

[9]Statisticians use the lower case Greek letter σ (sigma) to designate the standard deviation of a random variable. Since σ cannot be observed directly, the best we can do is to estimate its value from the measurements we make. As we shall see shortly, there are several ways to do this. Strictly speaking, the symbol for a computed estimate (a statistic) should be different from the symbol used for the parameter itself. The most commonly used symbol today for an estimated standard deviation is $\hat{\sigma}$. When working with control charts, though, the tradition is to use the simpler notation, even though this can be confusing.

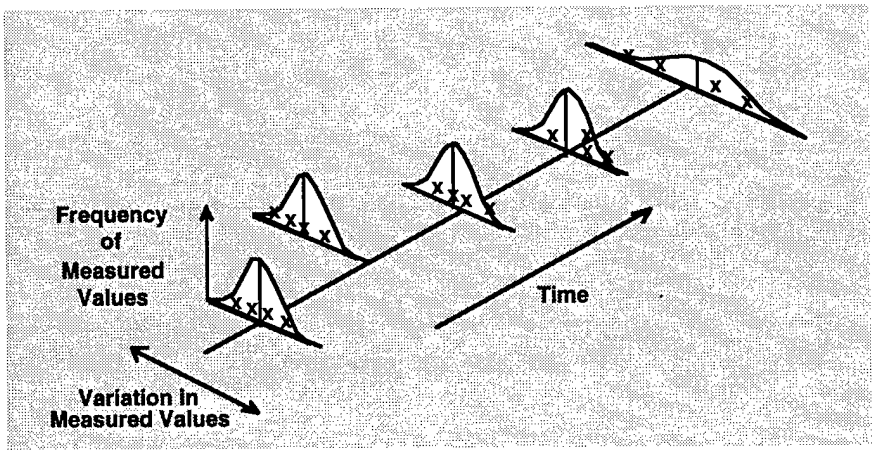


Figure 5-5: An Idealized Out-of-Control Process

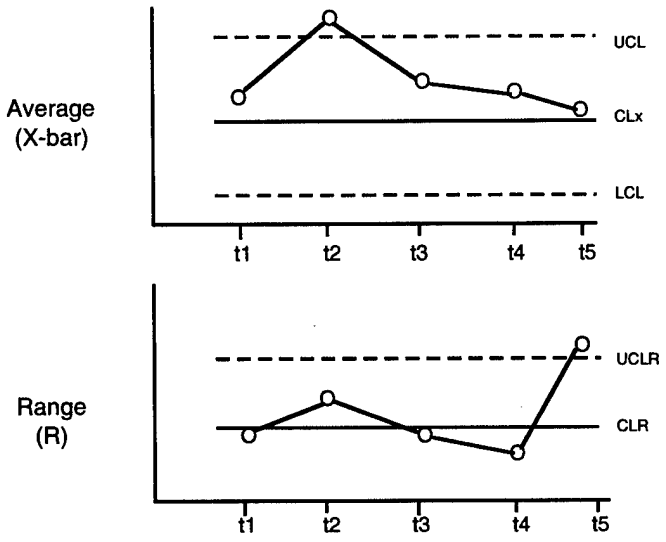


Figure 5-6: X-Bar and R Charts for an Out-of-Control Process

The Structure of Control Charts

> *One sees in practice countless control charts, most of which are unfortunately used incorrectly. It is to be feared that many of them do more harm than good. A necessary requirement for successful use of a control chart is a smattering of knowledge of the theory behind it.*
>
> *W. Edwards Deming, 1986*

There is more to preparing and using control charts than selecting a chart type, plotting data, and calculating control limits. While we will try to alert you to some of the factors that should be considered, if you are not currently conversant in the use of control charts, you should consult one of the many texts available on the subject. The books of Wheeler, Pyzdek, and Montgomery are particularly good at describing how to construct and use control charts [Wheeler 92, 93, 95; Pyzdek 90, 92; Montgomery 96]. These references are well grounded in theory, but their real focus is on understanding and applying basic principles and avoiding pitfalls. They do not require an educational background steeped in statistics or advanced mathematics.

To begin, all classical control charts have a center line and control limits on both sides of the center line. Both the center line and the limits represent estimates that are calculated from a set of observations collected while the process is running. The center line and control limits cannot be assigned arbitrarily, since they are intended to reveal what the process can actually do (its current level of performance), not what you or anyone else wants it to do.

The values plotted on a control chart can be values obtained from any statistic that has been defined on the sequence of individual measurements.[10] Subgroup averages, subgroup ranges, moving ranges, and individual values themselves are all examples of statistics that can be plotted in time sequence and used as a basis for control charts.

The basic layout of control charts is illustrated in Figure 5-7. The center line is usually the observed process average, although sometimes other measures of central tendency such as the median or mid-range are used. The control limits on either side of the center line are derived from one of several possible measures of process variability—ranges within subgroups being probably the most common.[11] The traditional (Shewhart) control limits are

---

[10] A statistic is a single-valued function of observable random variables, which is itself an observable random variable and which contains no unknown parameters [Mood 74].

[11] Whatever the method for estimating control limits, you should always try to ensure that the estimates are based solely on the effects of common cause variation. Any special-cause variations that creep in will make the limits too wide.

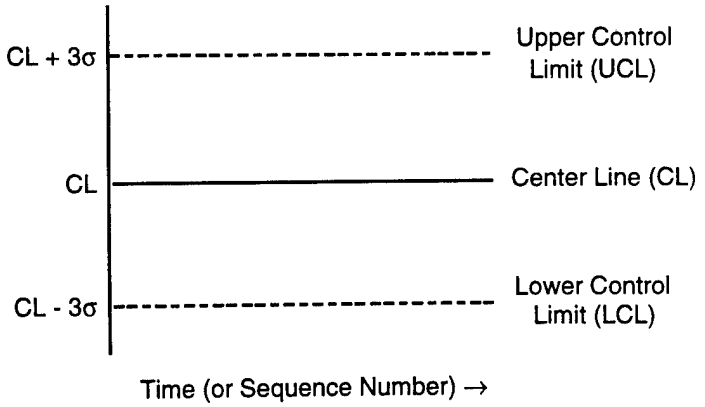


Figure 5-7: Control Chart Structure

±3 sigma, where sigma is the (estimated) standard deviation of the statistic plotted on the chart.[12] When a control limit lies beyond the range of possible outcomes, such as a negative value for product size, number of defects, or subgroup range, that control limit is usually omitted.

Both Shewhart and Wheeler make strong cases for always using 3-sigma limits for control charts [Shewhart, 31, 39; Wheeler 92, 95]. Using 3-sigma limits avoids the need to make assumptions about the distribution of the underlying natural variation. Moreover, experience over many years of control charting has shown 3-sigma limits to be economical in the sense that they are adequately sensitive to unusual variations while leading to very few (costly) false alarms—regardless of the underlying distribution.

Wheeler also shows how easy it is to estimate sigma incorrectly if you don't have a firm grasp on the concepts of rational sampling and rational subgrouping. For instance, people not well versed in control charting often use the standard deviation across all observed values as their estimate for sigma. This erroneous procedure lumps assignable cause variation together with common cause variation and almost always leads to control limits that are far too wide to be effective. Wheeler does an excellent job of illustrating these points. We urge you to read his books.

Since the values you plot on control charts are statistics, they can be any function of the measured properties of a product or process that you wish (or have reason to expect) to be constant over time. These data can be obtained by measuring attributes such as team size,

---

[12] On X-bar charts, sigma will be the standard deviation for individual values divided by $\sqrt{n}$, where n is the number of observations in each subgroup. Thus,

$$sigma_{\bar{x}} = \frac{sigma_x}{\sqrt{n}}.$$

elapsed time between milestones or events, staff hours expended per task, productivity, personnel turnover, tardiness, tool usage, backlog, number of defective units found, number of defects found per 1000 units, and so forth.

In software environments, as in many other white-collar activities, measurements often occur only as individual values. You may not have opportunities to draw samples of size $n > 1$ in ways where the inherent variation within each sample can reasonably be assumed to be constant. As we shall see shortly, this often leads us away from X-bar and R charts and toward a preference for using individuals and moving range (XmR) charts for examining the time-sequenced behavior of process data. We will illustrate the procedures for constructing and interpreting XmR charts later in this chapter.

Control charts have been used with success in industry since the 1920s, both in the United States and in other countries. As Montgomery points out, there are at least five reasons for their popularity [Montgomery 96]:

1. **Control charts are a proven technique for improving productivity.** When used properly, control charts reduce scrap and rework—the primary killers in any operation.

2. **Control charts are effective in defect prevention.** Control charts help keep the process in control, which is consistent with "do it right the first time" philosophies. When a process is not in control, you are not only paying someone to make a nonconforming product, but you must then rely on costly inspections to separate good product from bad to avoid dissatisfied customers.

3. **Control charts prevent unnecessary process adjustments.** They do this by helping you distinguish between background noise and abnormal variation. No other device, including a human operator, is as effective in making this distinction. Without control charts, it is all too easy to overreact to background noise and overcontrol the process. Making unnecessary adjustments to a process always increases the variability in results.

4. **Control charts provide diagnostic information.** Patterns found in control charts often contain diagnostic information that points to likely causes of abnormal behavior. This information greatly facilitates identifying and making the right changes.

5. **Control charts provide information about process capability.** They provide information about both the values of important process parameters and their stability over time. This provides a rational basis for predicting future performance, which is of immense use to product and process designers.

The Distinction Between Variables Data and Attributes Data

When measurements are used to guide process management and improvement, they are traditionally viewed as falling into one of two classes: *variables data* or *attributes data*. Control limits for attributes data are often computed in ways quite different from control limits for variables data. Unless you have a clear understanding of the distinctions between the two kinds of data, you can easily fall victim to inappropriate control charting methods.

Variables data (sometimes called measurement data) are usually measurements of continuous phenomena. Familiar examples from physical environments include measurements of length, weight, height, volume, voltage, horsepower, torque, efficiency, speed, and viscosity. In software settings, elapsed time, effort expended, years of experience, memory utilization, cpu utilization, and cost of rework would be all considered examples of variables data.

Attributes data, on the other hand, have a different origin and serve a different purpose. They occur when information is recorded only about whether an item conforms or fails to conform to a specified criterion or set of criteria. Attributes data almost always originate as counts—the number of defects found, the number of defective items found, the number of source statements of a given type, the number of lines of comments in a module of *n* lines, the number of people with certain skills or experience on a project or team, the percent of projects using formal code inspections, and so forth.

Because attributes data arise from counts and continuous phenomena yield variables data, several authors (some of our favorites included) have fallen into the trap of equating variables data to continuous phenomena and attributes data to counts. Nothing could be further from the truth. There are many situations where counts get used as measures of size instead of frequency, and these counts should clearly be treated as variables data. Examples include counts of the total number of requirements, total lines of code, total bubbles in a data-flow diagram, McCabe complexities, customer sites, change requests received, backlogged items, and total people in an organization or assigned to a project. When we count things like these, we are counting all the entities in a population, not just the occurrences of entities with specific attributes. Counts of entities that represent the size of a total population should almost always be treated as variables data, even though they are instances of discrete counts.

The key to classifying data as attributes data or variables data, then, depends not so much on whether the data are discrete or continuous as on how they are collected and used. For example, the total number of defects found, although a count based on attributes, is often used as a measure of the amount of rework and retesting to be performed. When used this way, most people would view a total count of defects as a measure of size and treat it as variables data, especially when it comes to choosing models for analysis and methods for computing control limits. Similarly, the number of working days in a month might properly be viewed as attributes data (if used as a numerator to compute the proportion of a month that

is available for working) or as variables data (if used as a denominator to normalize some other measure of activity or frequency of occurrence).

In short, the method of analysis that you choose for any data you collect will depend on the questions you are asking, the probability model that you have in mind, and the assumptions that you are willing to make with respect to the nature of the data. We will illustrate these issues at several points in the discussions that follow.

Two cautionary notes: as Wheeler and Chambers point out, one of the keys to making effective use of attributes data lies in preserving the ordering of each count in space and time. Sequence information (the order in time or space in which the data are collected) is often needed to correctly interpret counts of attributes. When people think of attributes data as simple tallies, they often fail to impose a useful structure on their data. This failure is a major cause of disappointing results in the use of attributes data [Wheeler 92].

Wheeler and Chambers also stress the importance making counts specific. This is not a major problem with continuous data, because the act of measuring usually makes the values specific enough. But with attributes data and other counts, the act of counting easily allows vagueness to creep in. There must be an operational definition (a clear set of rules and procedures) for recognizing an attribute or entity if what gets counted is to be what the user of the data expects the data to be. Checklists like the ones illustrated in Section 3.2 of this guidebook and in the SEI reports on defining software measures are examples of structured approaches that can be used to address this need [Park 92, Goethert 92, Florac 92].

Detecting Instabilities and Out-of-Control Situations

> *The basic assumption underlying most statistical techniques is that the data are a random sample from a stable probability distribution, which is another way of saying that the process is in statistical control. It is the validity of this basic assumption which the control chart is designed to test.*
>
> *Mary G. Natrella, 1966*

> *...in developing a control criterion we should make the most efficient use of the order of occurrences as a clue to the presence of assignable causes.*
>
> *Walter A. Shewhart, 1931*

To test for instabilities in processes, we examine control charts for instances and patterns that signal nonrandom behavior. Values falling outside the control limits and unusual patterns within the running record suggest that assignable causes exist.

Several tests are available for detecting unusual patterns and nonrandom behavior. The Western Electric handbook cites four that are effective [Western Electric 58, Wheeler 92, 95]:

Test 1: A single point falls outside the 3-sigma control limits.

Test 2: At least two of three successive values fall on the same side of, and more than two sigma units away from, the center line.

Test 3: At least four out of five successive values fall on the same side of, and more than one sigma unit away from, the center line.

Test 4: At least eight successive values fall on the same side of the center line.

Tests 2, 3, and 4 are based on the presumptions that the distribution of the inherent, natural variation is symmetric about the mean, that the data are plotted in time sequence, and that successive observed values are statistically independent. The symmetry requirement means that the tests are designed primarily for use with X-bar and individuals charts. Strictly speaking, they are not applicable to R charts, S charts, or moving range charts. However, as the Western Electric handbook shows, you will not go far astray if you use tests 2, 3, and 4 for R charts that are based on samples of size 4 or 5. For samples of size 2, however, the rules should be modified to account for the decidedly nonsymmetric nature of the range distribution. If you want to use these kinds of tests with small samples, you will find the modified rules described on page 182 of the Western Electric handbook [Western Electric 58].

When you use the four tests, process instability is indicated whenever one or more of the conditions exist. The effect of using all four tests together, compared to using test 1 alone, is to reduce the average number of points between false alarms in a controlled process from 370 to 91.25 [Champ 87].[13] Thus the increased sensitivity that you get for detecting assignable causes comes at the cost of a four-fold increase in false alarms.

As Wheeler points out, the four tests are a conservative and practical subset of the much larger body of tests that have been used from time to time in industrial settings [Wheeler 92, 95]. Many other tests are also possible. Descriptions of some of these tests and assessments of their properties can be found in several references [Montgomery 96, Hoyer 96, Western Electric 58, Champ 87]. Each additional test increases your chances of detecting an out-of-control condition. Unfortunately, it also increases your chances of getting a false alarm. In statistics, as in life, there is no free lunch.

When using additional tests to supplement 3-sigma limits, it is important to understand that any decision to use a test should, in principle, be made *before* looking at the data. Waiting

---

[13]The reference value of 370 was derived by assuming that the inherent variation is normally distributed. The number is different for other distributions.

until after you see the data to decide which patterns to treat as unusual has the effect of increasing the potential for false alarms, perhaps substantially.

The temptation to violate this guideline is strong, as the human mind has a marvelous ability to perceive patterns after the fact. An "I'll know it when I see it" attitude can be very useful when interpreting patterns to deduce what may have caused an unusual event, but it falls easily into the trap of circular reasoning if used to single out a previously undefined pattern as a signal of an unusual event.[14] Deming is quite explicit in pointing this out:

> *Search for a pattern can be overdone. It is necessary to state in advance what the rules are for indication of a special [assignable] cause. One can always concoct a pattern that will indicate anything desired, once the chart is in hand.*
>
> *W. Edwards Deming, 1986*

Some organizations, though, have very good reasons for identifying specific patterns that are likely to signal something going wrong. For example, you may know that when an assignable cause of type X occurs, a pattern of type Y often follows. If few other causes produce patterns of type Y, and if it is possible for assignable causes of type X to creep into the process, it makes sense to be on the lookout for patterns of type Y and to use their appearance as a signal for action. It would be wise, however, to chart the effectiveness of your detection rule in order to determine the frequency with which it leads to false alarms. Your guidelines for using the rule can then be tailored accordingly.

The Stability Investigation Process

> *Statistical control [is] not mere application of statistics... Classical statistics start with the assumption that a statistical universe exists, whereas control theory starts with the assumption that a statistical universe does* ***not*** *exist.*
>
> *Walter A. Shewhart, 1943*

Figure 5-8 shows the steps involved in investigating the stability of a process. It provides a context for discussing the use of product or process measurements to evaluate process stability.

---

[14]Excellent discussions and illustrations of patterns and what they may tell you can be found in Montgomery's text and in the Western Electric Handbook [Montgomery 96, Western Electric 58].

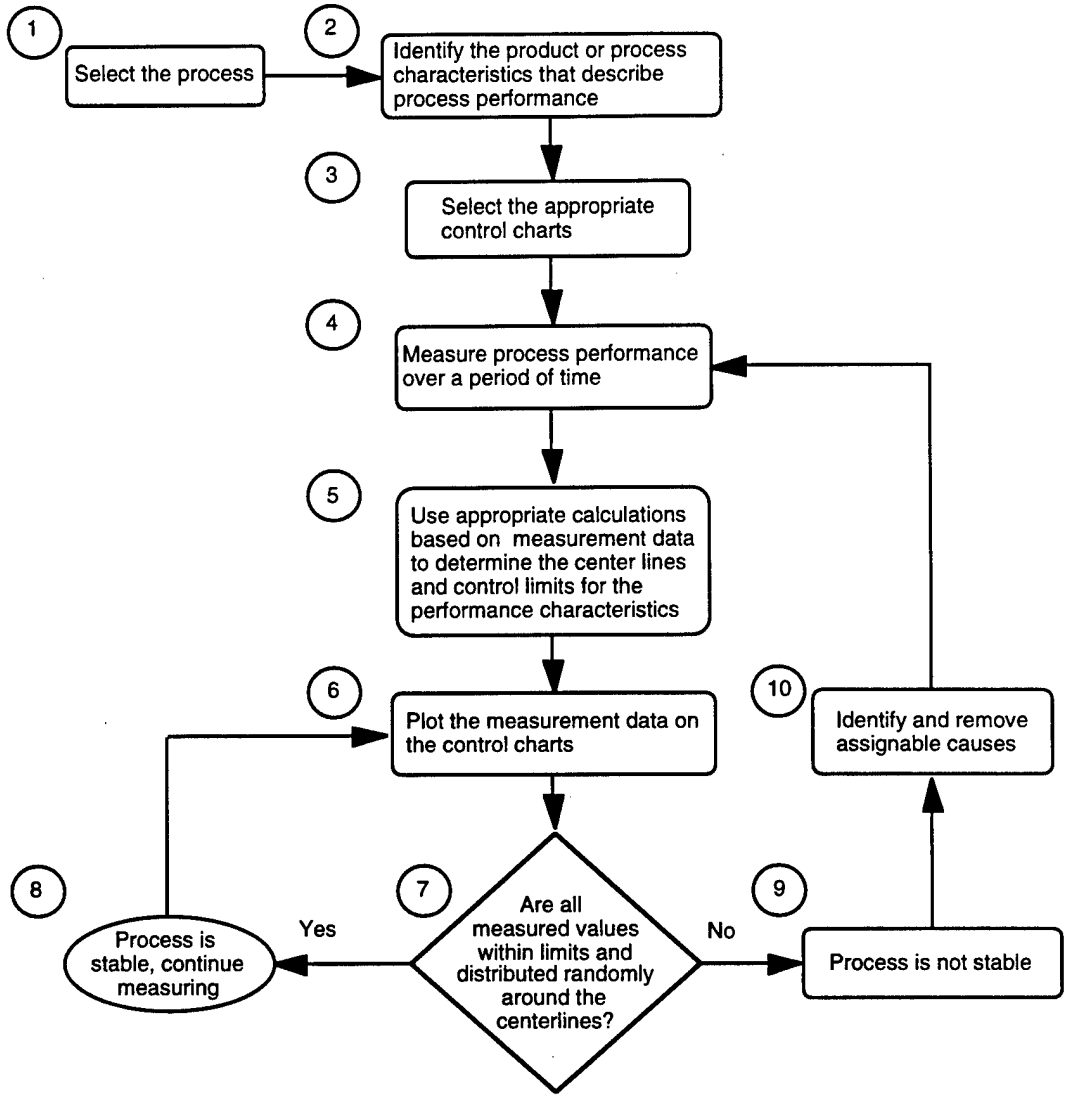


Figure 5-8: The Steps for Using Control Charts to Evaluate Process Stability

The steps in Figure 5-8 are as follows:

1. Select the process to be evaluated for stability.
2. Identify the product or process characteristics that describe process performance.
3. Select the appropriate types of control chart.
4. Measure product and/or process characteristics over a period of time. The number of measurements and the period will be a function of the process and will depend on the type of control chart used.

5. Use appropriate calculations, applied to the measurement data, to establish the center lines and limits of variation for normal process performance. The specific calculations will vary depending on the type of control chart used.
6. Plot the measurement data obtained from Step 4 on the control charts.
7. Compare the values plotted in Step 6 to the center-line values and limits established in Step 5.
8. If all plotted values are distributed randomly above and below the center lines and within the control limits, conclude that the process has been stable at least for the period or events covered by the measurements. Continue measuring and plotting the data to ensure that the process remains stable.
9. If any plotted value exceeds the limits, or if the pattern of values exhibits other nonrandom behavior, conclude that the process is not stable. The reasons for such observations must be investigated. If an assignable cause is found and the result is detrimental, fix the process so that it cannot happen again. If the result is beneficial, try to make the cause part of the process.
10. Once all assignable causes have been removed, the control limits should be recalculated. This may mean that you will need additional observations to determine reliable limits for the process as corrected.

If, by following the above steps, you find that the process appears to be in control, and if you have grounds for believing that the process will continue to operate in the future as it has in the past, the limits found in Step 5 may be used to predict future process behavior.

On the other hand, if you find that the process is not stable, the limits derived in Step 5 are not valid for predicting process behavior; they merely pass judgment on the past performance of the process. Once the assignable causes have been corrected, the procedure for determining limits must start again at Step 4.

Even when out-of-control data are included in the values used to compute the control limits at Step 5, effective (rational) subgrouping will ensure that the limits are almost always sufficiently tight to identify lack of control. Using all the data as one large group to compute an estimate for sigma is almost always wrong, because it presumes that the process is in control (i.e., that a single, stable distribution exists). Limits computed from the full data set can easily be much too wide because they are contaminated by assignable cause variations. The purpose of rational subgrouping is to minimize the effects of assignable cause contamination. Wheeler illustrates these points effectively in his books [Wheeler 92, 95]. We will discuss rational subgrouping in more detail in Section 7.6.

The use of stability criteria to assess the predictability of process performance places this topic squarely in the setting of an analytic study. All the cautions and caveats that will be

discussed in Section 7.1 apply. Nevertheless, demonstrating that a process has historically behaved in a controlled and stable way provides the most reasonable basis possible for extrapolating to the future. Just think of the alternative: if you knew that your process was unstable, how much faith would you place in your ability to predict its future performance?

5.3 Control Charts for Variables Data

The control chart is the process talking to us.

Irving Burr, 1953

To see how control charts can be used to investigate process stability, we will look at four different kinds of charts:

- X-bar and range charts
- individuals and moving range charts (often called XmR charts)
- u charts
- Z charts

The first two kinds of charts are illustrated in this section. The third and fourth charts (and XmR charts, again) will be illustrated when we address control charts for attributes data in Section 5.5.

X-Bar and Range Charts

Charts for averages, often called X-bar charts, are used with subgrouped data. They apply when measurements for subgroups of size $n \geq 2$ can be obtained under conditions where the variation can reasonably be expected to be unchanging within each subgroup. Although these conditions seem to occur less frequently in software settings than in industrial manufacturing, you should be familiar with the capabilities of charts that can take advantage of this kind of subgrouping. X-bar and range charts can be very useful when sampling opportunities are plentiful and assumptions of homogeneous variation can be supported.

X-bar charts are used to answer the question, "Has the central tendency of the process changed over the time period observed?" The corresponding range charts (often called R charts) address the question, "Has the dispersion in the observed values been affected by assignable causes?" Charts for averages and ranges are used together to identify points where a process has gone out of control. Measurements of product or process characteristics are grouped into self-consistent sets (subgroups), and the results of the groupings are used to calculate the limits used to examine stability and control the process.

The procedure for calculating control limits for X-bar and R charts is shown in Figure 5-9 [Wheeler 92]. The terms A_2, D_3, and D_4 are conventional symbols for factors that have been tabulated by statisticians for converting averages of subgroup ranges into unbiased estimates for 3-sigma limits. An abbreviated table of these factors is shown in Figure 5-10. More extensive tables can be found in almost any book on statistical process control [Duncan 74, Wadsworth 86, Ott 90, Wheeler 92, Wheeler 95, Grant 96, Montgomery 96].

1. Compute the average and range for each of the k subgroups.
2. Compute the grand average, $\overline{\overline{X}}$, by averaging each of the k subgroup averages.
3. Compute the average range, $\overline{R}$, by averaging each of the k subgroup ranges.
4. The center line for the $\overline{X}$ chart is $\overline{\overline{X}}$. The center line for the R chart is $\overline{R}$.
5. Find the values for A_2, D_3, and D_4 which correspond to the subgroup size n. (See the table in Figure 5-10.)
6. Multiply $\overline{R}$ by A_2, giving $A_2\overline{R}$.
7. Add $A_2\overline{R}$ to the grand average to obtain the upper control limit for the $\overline{X}$ chart. $\{UCL_{\overline{X}} = \overline{\overline{X}} + A_2\overline{R}\}$
8. Subtract $A_2\overline{R}$ from the grand average to obtain the lower control limit for the $\overline{X}$ chart. $\{LCL_{\overline{X}} = \overline{\overline{X}} - A_2\overline{R}\}$
9. Multiply $\overline{R}$ by D_4 to get the upper control limit for the R chart. $\{UCL_R = D_4\overline{R}\}$
10. Multiply $\overline{R}$ by D_3 to get the lower control limit for the R chart. $\{LCL_R = D_3\overline{R}\}$

Figure 5-9: Procedure for Calculating Control Limits for X-Bar and R Charts

One caution: the procedure in Figure 5-9 uses the observed ranges within subgroups as the basis for estimating the standard deviation of the natural process variation. The efficiency of this method falls off rapidly as the size of the subgroups increases. For this reason, most experts advise that range charts be used only when there are less than 10 observations in each subgroup. For subgroups of size 4, 5, and 6, the procedure in Figure 5-9 works very well. For larger subgroups, S charts based on averages of quadratic estimators for the standard deviation within subgroups give more reliable results. S charts are also

n	A_2	D_3	D_4
2	1.880	—	3.268
3	1.023	—	2.574
4	0.729	—	2.282
5	0.577	—	2.114
6	0.483	—	2.004
7	0.419	0.076	1.924
8	0.373	0.136	1.864
9	0.337	0.184	1.816
10	0.308	0.223	1.777

Figure 5-10: Constants for Computing Control Limits for X-Bar and R Charts

preferred when the subgroup size is not constant. In these cases, computing an average range is meaningless, since the expected value for the range within each subgroup varies with subgroup size.

The steps for using S charts parallel those in Figure 5-9. However, the factors A_2, D_3, and D_4 get replaced with alternative factors that are consistent with the use of quadratic estimators for sigma. There are two ways to compute S, and hence two sets of factors. These factors and the steps for constructing S charts are discussed in many books that provide instruction on control charting. Montgomery, for example, provides excellent illustrations [Montgomery 96]. For economic reasons, it is usually wise to avoid subgroups that are larger than 5 or 6 even when using S charts. So, unless you enjoy calculating root sum squares, and as long as your subgroup sizes are constant, R charts will serve you well.

Example

John Smith, a software manager at Homogeonics, Inc. is responsible for developing the follow-on release to an existing product. He is also responsible for providing support service to users of the existing product. The development schedule for the new product has been based on the assumption that product support service will require about 40 staff hours per day until the new product is released. Because of the intermittent behavior and potential complexity of service requests, everyone on the development team must be available to provide support service at any given time. Mr. Smith is concerned that the daily effort to support service requests stays within the range assumed by the plan, for if the effort exceeds the plan for a sustained period, the development schedule will be in jeopardy, and alternative plans will be needed.

The record for product-support staff hours expended per day by the technical staff for the past 16 weeks is shown in Figure 5-11.

Week	Mon	Tues	Wed	Thur	Fri	Average	Range
1	50.5	43.5	45.5	39.8	42.9	44.44	10.7
2	44.3	44.9	42.9	39.8	39.3	42.24	5.6
3	48.8	51.0	44.3	43.0	51.3	47.68	8.3
4	46.3	45.2	48.1	45.7	44.1	45.88	4.0
5	40.6	45.7	51.9	47.3	46.4	46.38	11.3
6	44.4	49.0	47.9	45.5	44.8	46.32	4.6
7	46.0	41.1	44.1	41.8	47.9	44.18	6.8
8	44.9	43.4	49.0	49.5	47.4	46.84	6.1
9	50.0	49.0	42.6	41.7	38.5	44.36	11.5
10	44.5	46.5	41.7	42.6	41.7	43.40	4.8
11	43.8	41.8	45.5	44.5	38.6	42.84	6.9
12	37.2	43.8	44.8	43.5	40.9	42.04	7.6
13	50.0	43.4	48.3	46.4	43.4	46.30	6.6
14	52.3	45.2	42.2	44.8	42.8	45.46	10.1
15	50.0	46.2	47.4	42.2	47.0	46.56	7.8
16	47.3	49.7	48.0	42.0	41.0	45.60	8.7
Grand Averages						45.03	7.59

Figure 5-11: Hours of Product-Support Effort for a 16-Week Period

To examine the variability in the data, Mr. Smith grouped the values by week, computed the average daily hours for each week (each week is a subgroup), and used the results to compute control limits for the weekly averages and ranges. He then plotted the weekly averages, ranges, and limits on control charts to see if there were any signs of unstable conditions. The computations he used were as follows, and the results are shown in Figure 5-12.

The grand average is $\overline{\overline{X}} = 45.03$ staff hours.

The average range is $\overline{R} = 7.59$ staff hours.

The subgroup size is $n = 5$.

From the table in Figure 5-10: $A_2 = 0.577$, $D_3 = 0$, and $D_4 = 2.114$

From the formulas in Figure 5-9:

$$UCL_{\overline{X}} = \overline{\overline{X}} + A_2\overline{R} = 45.03 + 0.577(7.59) = 49.41$$

$$CL_{\overline{X}} = \overline{\overline{X}} = 45.03$$

$$LCL_{\overline{X}} = \overline{\overline{X}} - A_2\overline{R} = 45.03 - 0.577(7.59) = 40.66$$

$$UCL_R = D_4\overline{R} = 2.114(7.59) = 16.04$$

$$CL_R = \overline{R} = 7.59$$

$$LCL_R = D_3\overline{R} = \text{undefined (does not exist for } n = 5)$$

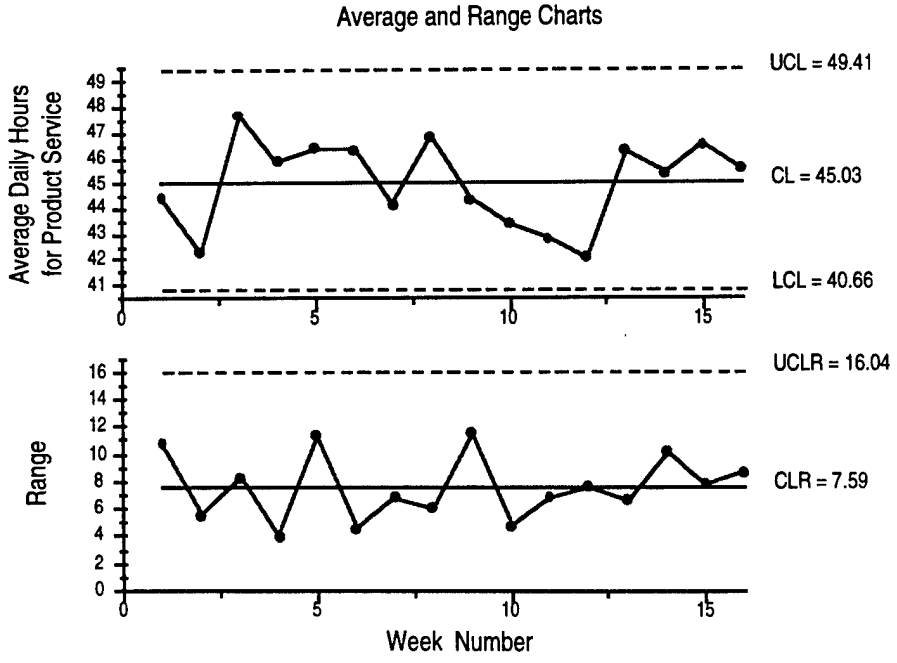


Figure 5-12: X-Bar and R Charts for Daily Product-Service Effort

The weekly averages and ranges plotted in Figure 5-12 are well within the control limits, thus meeting the primary requirement for statistical control (Shewhart's Criterion I). To test for other signs of instability, the manager also looked for patterns of runs in the data, as discussed in Section 5.2. The additional tests he used were

Test 2: At least two of three successive values fall on the same side of, and more than two sigma units away from, the center line.

Test 3: At least four out of five successive values fall on the same side of, and more than one sigma unit away from, the center line.

Test 4: At least eight successive values fall on the same side of the center line.

To perform these tests, he needed an estimate for $sigma_{\bar{X}}$, the standard deviation of a subgroup average. This estimate is given by

$$sigma_{\bar{X}} = \frac{A_2 \bar{R}}{3}$$

which, for the example, gives

$$sigma_{\bar{X}} = \frac{0.577(7.59)}{3} = 1.46 \text{ product - service hours.}$$

Mr. Smith used this information to plot lines at 1-sigma intervals on the chart for average daily product-service hours. Figure 5-13 shows the result. With the added guidelines, he could see that none of the patterns described in tests 2–4 were present. He concluded, therefore, that the staff-hour expenditures for product service are consistent with a stable process, and that the average daily value is approximately five hours per day above the planned value. Since no instances of assignable cause variation are apparent, he knows he will need to change the product-service process (or the quality of the product) if he wants to reduce the variation or center it about his planning target.

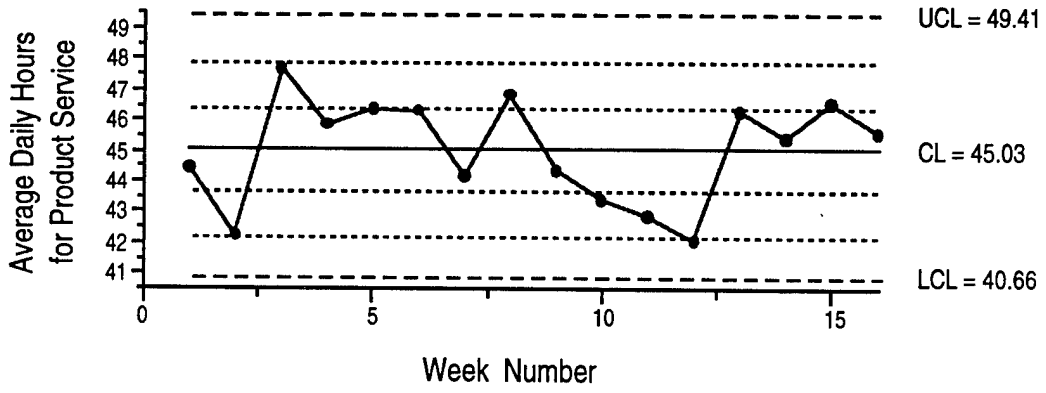


Figure 5-13: Testing for Patterns in Weekly Product-Service Data

Individuals and Moving Range Charts for Continuous Data

When measurements are spaced widely in time or when each measurement is used by itself to evaluate or control a process, a time-sequenced plot of individual values rather than averages may be all that is possible. The subgroup size n is then 1, and the formulas based on subgroup ranges that are used for plotting limits for X-bar and R charts no longer apply, so another way is needed to calculate control limits. Here the principle of rational subgrouping comes into play, and we use the short-term variation between adjacent observed values to estimate the natural (inherent) variation of the process. This leads to a pair of charts, one for the individual values and another for the successive two-point moving ranges. This combination of charts for individual observations and moving ranges is called an XmR chart, where X and mR symbolize the individual values and moving range, respectively.[15]

The idea behind XmR charts is that, when subgroups can easily include nonrandom components, we minimize the influence that nonrandom effects have upon estimates for sigma by keeping the subgroups as small as possible. The smallest possible subgroup size is 1. There is no way to estimate sigma from a single measurement, so we do the next best thing—we attribute the changes that occur between successive values to the inherent variability in the process. The absolute values of these changes are called two-point moving ranges.

When m sequential measurements are available, we will have $m-1$ moving ranges. If one of these moving ranges is contaminated with an assignable cause, we will still have $m-2$ moving ranges that are representative of the inherent variation. Thus XmR charts are essentially a strategy for isolating assignable causes. When we average the moving ranges to derive an estimate for sigma, some error may be present, but 1 large range out of 20 or so will not have an unacceptably large effect on the results.

Factors exist that enable us to calculate 3-sigma limits for both individual values and the moving ranges. The following formulas apply:

The kth (two-point) moving range $\quad mR_k = |X_{k+1} - X_k|$

Average moving range $\quad \overline{mR} = \frac{1}{m-1}\sum_{1}^{m-1} mR_k$

where m is the total number of data points.

---

[15]The Western Electric handbook gives these examples for the principal kinds of data for which XmR charts are applicable [Western Electric 58]:

- accounting figures of all kinds, including shipments, efficiencies, absences, losses, inspection ratios, maintenance costs, accident reports, records of tests, etc.
- production data such as temperatures, pressures, voltages, humidity, conductivity, furnace heat, gas consumption, the results of chemical analysis, etc.

Center line (the average of individual values)	$CL_x = \overline{X} = \frac{1}{m}\sum_{1}^{m} X_k$
Upper natural process limit	$UNPL_x = \overline{X} + 2.660\overline{mR}$
Lower natural process limit	$LNPL_x = \overline{X} - 2.660\overline{mR}$
Center line or average moving range	$CL_R = \overline{mR}$
Upper control limit for moving range	$UCL_R = D_4\overline{mR} = 3.268\overline{mR}$
Lower control limit for moving ranges	LCL_R does not exist for $n = 2$

The 3-sigma distance associated with individual values above is given by the formula

$$3sigma_X = \frac{3\overline{mR}}{d_2} = 2.660\overline{mR}$$

So we can estimate $sigma_X$ as follows:

$$sigma_X = \frac{\overline{mR}}{d_2}$$

The subgroup size for two-point ranges is $n = 2$, so the value for D_4 obtained from Figure 5-10 is $D_4 = 3.268$. The value for d_2 is obtained similarly from a table of dispersion adjustment factors that converts average ranges to estimates for sigma.[16,17] An abbreviated form of this table, shown in Figure 5-14, shows that $d_2 = 1.128$.

subgroup size n	d_2
2	1.128
3	1.693
4	2.059
5	2.326
6	2.534
7	2.704
8	2.847
9	2.970
10	3.078

Figure 5-14: Dispersion Adjustment Factors for Range Data

[16]The values for d_2 have been computed by Harter [Harter 60]. They can be found tabulated for a wider range of n values in most references that provide instruction in control charting methods.

[17]Strictly speaking, the values tabulated for d_2 assume that the ranges have been averaged for a fairly large number of subgroups, say 20 or more. When only a few subgroups are available, the values for d_2 are somewhat too small, and limits computed by applying d_2 to the average range will be too wide. If this is of concern to you, you may wish to consult the table of corrected values d_2^* tabulated by A. J. Duncan. This table appears on page 950 of his book and on page 417 of Wheeler's 1995 work [Duncan 74, Wheeler 95]. In most applications of control charts, however, it suffices to use estimates for sigma that are based on d_2.

Although the impracticality of grouping may be one reason for charting individual values, there are other reasons that can motivate you to plot individual values voluntarily. For example, the Western Electric handbook lists five types of conditions that may be detected more readily with individuals charts than with X-bar and R charts [Western Electric 58]:

1. cycles (regular repetitions of patterns)
2. trends (continuous movement up or down)
3. mixtures (presence of more than one distribution)
4. grouping or bunching (measurements clustering in spots)
5. relations between the general pattern of grouping and a specification

XmR charts can also occur as a natural evolution of simple run charts, once sufficient points become available to compute reasonably reliable control limits.

Care should always be exercised when interpreting patterns on a moving range chart. Moving ranges are correlated, and this correlation can induce patterns of runs or cycles. For this reason, some authorities recommend not plotting the moving range chart at all [Roes 93, Rigdon 94]. They point out also that the moving range cannot really provide any useful information about a shift in process variability that is not apparent in an individuals chart alone. However, as long as little weight is given to patterns like runs and cycles in a moving range chart, little harm will come from constructing the plot, and it may help people who are accustomed to X-bar and R charts feel comfortable with your results. Once a process is in control, though, the false alarm rate when using two charts together is likely to be roughly three times as large as when using an individuals chart alone [Rigdon 94].

Example

Mr. Smith, the software manager at Homogeonics, Inc., is interested in evaluating the day-to-day variation of effort spent on servicing the existing product. He suspects that different factors may influence activity levels on different days of the week and that there may be differences from week to week. Because day-to-day variation lies at the heart of his question, grouping the data would not be rational. XmR charts that plot the data in the order in which they are obtained then become the appropriate charts to use.

The 80 data points in Figure 5-11 provide a sequence that we can use to construct the individuals and moving range charts. From the data, we find the average staff hours per day ($\overline{X}$) and the average two-point moving range ($\overline{mR}$) to be

$$\overline{X} = 45.03 \text{ and } \overline{mR} = 3.38$$

The computations then proceed as follows:

Center line (average of individual values) $CL_x = \overline{X} = 45.03$

Upper natural process limit	$UNPL_x = \overline{X} + 2.660\overline{mR} = 54.01$
Lower natural process limit	$LNPL_x = \overline{X} - 2.660\overline{mR} = 36.05$
Center line or average moving range	$CL_R = \overline{mR} = 3.38$
Upper control limit for moving range	$UCL_R = D_4\overline{mR}$ $= 3.268\overline{mR} = 11.03$
Sigma for individual values	$sigma_X = \frac{\overline{mR}}{d_2} = \frac{3.38}{1.128} = 2.996$

The results of these calculations are plotted with the data on the control charts in Figure 5-15. There are no out-of-limit points and no patterns that suggest unusual behavior. The process appears to be in statistical control from the perspective of individual values, just as it did when weekly averages were plotted on X-bar and R charts.[18]

Since nothing unusual is apparent in Figure 5-15, we can use the data to illustrate the concept of natural process limits. That discussion takes place shortly, in Section 5.4.

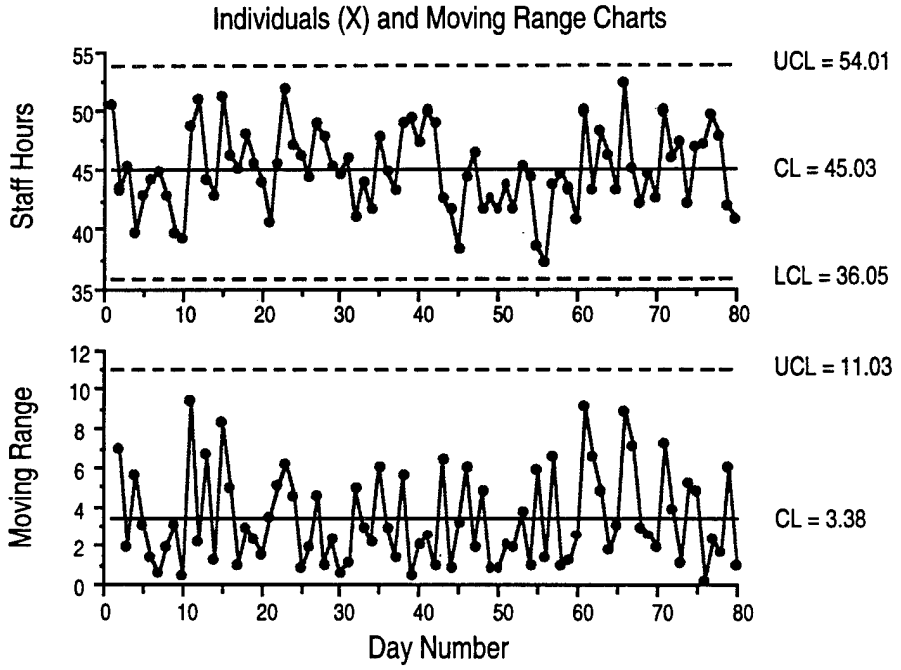


Figure 5-15: XmR Charts for Daily Customer-Service Effort

[18]The fact that group averages were in control was no guarantee that the individual values would be in control.

Individuals and Moving Range Charts for Discrete Data

Variables data are not always measured on continuous scales. Sometimes they take on only discrete values, such as when counts of objects are used to describe the size of an entity or the status of an activity. If the discrete measurements range over at least five distinct values, the data can usually be treated as if they were continuous. The usual X-bar, R, and XmR charts then apply. The following example illustrates this point.

Example

Each week a system test organization reports the number of critical problems that remain unresolved. The number of unresolved problems (URPs) are compared to the planned number, the average number unresolved since testing began, and the average number of unresolved problems in the previous weeks of the current quarter. At the conclusion of week 31, the weekly report contains the data shown in Figure 5-16.

System Test Report for Week 31

Number of unresolved problems (URPs)	Planned # of URPs	Deviation from planned value	Average URPs prior to current week	Deviation from grand average	Average URPs this quarter	Deviation from quarter average
28	19	+ 47.4%	20.13	+39.07%	20.50	+36.6%

Figure 5-16: Report of Unresolved Problems

The system test manager is concerned that something may have happened to the problem resolution process in week 31, and that the schedule for completing testing may now be threatened by something that is causing a surge in the backlog of unresolved problems. However, he is not totally convinced that the most recent value (28) is a signal that something is amiss. To help determine whether or not the current week's value may be simply the result of normal random variation, the history of unresolved problem reports for the current year has been obtained and tabulated. This history is shown in Figure 5-17.

History of In-Process Inventory of Unresolved Problem Reports

Week	1	2	3	4	5	6	7	8	9	10	11	12
1st Qtr	19	27	20	16	18	25	22	24	17	25	15	17
2nd Qtr	20	22	19	16	22	19	25	22	18	20	16	17
3rd Qtr	20	15	27	25	17	19	28					

Figure 5-17: Weekly History of Unresolved Problem Reports

This tabulation shows that on two previous occasions the number of unresolved problems approached the value of 28 in the latest week. But this doesn't fully resolve the issue for the

system test manager. Because the problem resolution process had seemed to be stable over the first two quarters, he decided to construct a control chart to see if it might point to any problems with the process. Since only one data point was obtained each week, and since week-to-week variation is the issue that concerns him, the most rational subgrouping for examining the inherent variability in the backlog data is the two-point moving range. XmR charts are then the appropriate tools for examining the issue.

The approach the test manager elected to take was to use the data from the first 2 quarters (24 weeks) to create the control chart. He then extended the chart by plotting the current quarter's data and holding the control limits constant. His computations, based on data from the first 24 weeks of testing, are shown below. As with XmR charts for continuous data, the average of the moving ranges were used to estimate the standard deviation of the inherent variation. The factors D_4 and d_2 that correspond to $n = 2$ (the subgroup size for two-point moving ranges) were obtained from the tables in Figures 5-10 and 5-14. These factors are 3.268 and 1.128, respectively.

$\overline{X} = 20.04$ for the first 24 weeks

$\overline{mR} = 4.35$

Center line (average of individual values)	$CL_x = \overline{X} = 20.04$
Upper natural process limit	$UNPL_x = \overline{X} + \frac{3\overline{mR}}{d_2} = \overline{X} + 2.660\overline{mR} = 31.6$
Lower natural process limit	$LNPL_x = \overline{X} - 2.660\overline{mR} = 8.49$
Center line (average moving range)	$CL_R = \overline{mR} = 4.35$
Upper control limit for moving range	$UCL_R = D_4\overline{mR} = 3.268\overline{mR} = 14.2$

The completed XmR charts for the backlog of critical problem reports, with the points for the third quarter added, are shown in Figure 5-18. These charts show no sign that anything unusual has happened (using all four tests for instability described earlier). Thus, the process appears to be stable, and the occurrence of 28 unresolved problems in week 31 is not a significant signal that the process has changed. Occasional random backlogs of size 28 (or even more) fall within the limits of natural variability that represent the voice of the process. The manager concluded that it would be unwise to take precipitous action to rectify the backlog reported in week 31.

Since the charts in Figure 5-18 show no signs of instability, the control limits reflect what the process is likely to do, as long as it continues to operate consistently and its inputs remain as consistent as they have in the past. If the manager wants to reduce either the average backlog or the amount of variation, the process itself will have to be changed.

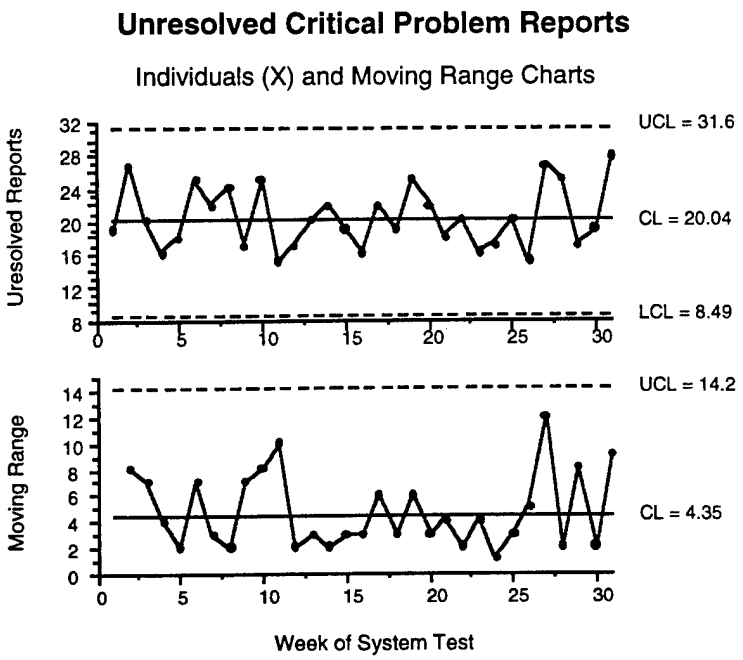


Figure 5-18: XmR Charts for Unresolved Problem Inventory

5.4 Frequency Histograms and Natural Process Limits

Constant systems of chance causes give rise to frequency distributions, often called statistical laws.

Walter A. Shewhart, 1931

The results of measurements are often used to construct histograms that summarize the historical performance of a process. Figure 5-19 is an example. Here the data from Homogeonics, Inc. have been grouped to obtain the frequency counts depicted by the vertical bars. The center line and control limits from the XmR charts have been added to the histogram to help characterize the data.

One caution: empirical distributions of the sort shown in Figure 5-16 can be misleading if the process that produced the data is not in statistical control. Histograms obtained from sequences of measurements invariably get interpreted as predictive of future performance. But it is impossible to say what a histogram of an out-of-control process represents. This means that a control chart for individual observations should always be examined before presenting a histogram of process results to anyone. Many erroneous conclusions have been drawn, even by veteran statisticians, because the fundamental assumption of a constant system of chance causes was not tested.

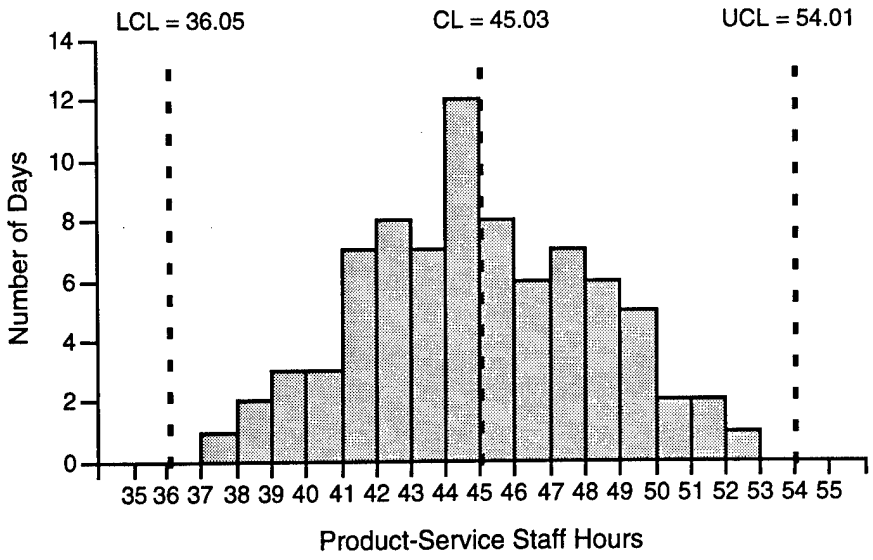


Figure 5-19: A Histogram of Measurements from a Stable Process

In the Homogeonics example, our investigations showed no signs of out-of-control behavior, so we may be justified in interpreting the data as having come from a single, constant system of chance causes.[19] Fluctuating or shifting distributions and mixtures of distributions, had they been present in significant size, could easily have produced unusual values or patterns. No histogram or control chart would then have predictive merit. It is only when control charts suggest that a process is stable, and we can assure that nothing related to the process will change, that we have a reasonable basis for using histograms like Figure 5-19 to characterize expectations of future performance.

When you have a stable process, the limits associated with a control chart or histogram for individual values describe what the process is able to do, as long as it continues to operate as it has in the past. These limits are called *natural process limits* because they represent the degree of variation inherent in the process once all assignable causes have been removed and prevented from recurring.

Natural process limits always describe the variability associated with individual measurements, not the variability of the subgroup averages used to construct X-bar charts.

---

[19]We say "may be" because it would help to know something about the characteristics and frequencies of incoming requests and the repeatability of the process that assigns people to service the requests.

5.5 Control Charts for Attributes Data

Thus far our discussions on using control charts to analyze process stability have focused on *variables data.* These data usually consist of observations of continuous phenomena or of counts that describe size or status. Counts related to occurrences of events or sets of characteristics, on the other hand, are traditionally referred to as *attributes data.* Examples include the number of defects in a module or test, the percentage of people with specified characteristics, the number of priority-1 customer complaints, and the percentage of nonconforming products in the output of an activity or process.

When attributes data are used for direct comparisons, they must be based on consistent "areas of opportunity" if the comparisons are to be meaningful. For example, if the number of defects that are likely to be observed depends on the size of a module or component, all sizes must be nearly equal. The same holds true for elapsed times if the probabilities associated with defect discovery depend on the time spent inspecting or testing.

In general, when the areas of opportunity for observing a specific event are not equal or nearly so, the chances for observing the event will differ across the observations. When this happens, the number of occurrences must be normalized (i.e., converted to rates) by dividing each count by its area of opportunity before valid comparisons can be made.

Although constant areas of opportunity occur fairly often in manufacturing situations, where product sizes and complexities are often relatively consistent from item to item, conditions that make us willing to assume constant areas of opportunity seem to be less common in software environments. This means that normalization will almost always be needed when using attributes data to evaluate software products and processes.

When areas of opportunity can change from observation to observation, measuring and recording the size of each area is something that must be addressed when you define procedures for counting the occurrences of events that interest you. For example, if defects are being counted and the size of the item inspected influences the number of defects found, some measure of item size will also be needed to convert defect counts to relative rates that can be compared in meaningful ways. Similarly, if variations in the amount of time spent inspecting or testing can influence the number of defects found, these times should be clearly defined and measured as well. If both size and inspection time can influence the number of defects found, models more elaborate than simple division will be needed to normalize defect counts so that they can be plotted meaningfully on control charts.

Distributional Models and Their Relationships to Chart Types

To set the stage for the discussions that follow, it helps to have an understanding of the different kinds of control charts that are used with attributes data. Figure 5-20 lists the most frequently used chart types [Wheeler 92]. Each type of chart is related to a set of assumptions (i.e., a distributional model) that must hold for that type of chart to be valid.

There are six types of charts for attributes data—np, p, c, and u charts, as well as XmR charts for counts and XmR charts for rates.

Data characterized by a binomial model		Data characterized by a Poisson model		Other data based on counts	
Area of Opportunity		Area of Opportunity		Area of Opportunity	
n constant	n variable	constant	variable	constant	variable
np chart or XmR	p chart or XmR	c chart or XmR	u chart or XmR	XmR charts for counts	XmR charts for rates

Figure 5-20: Table of Control Charts for Attributes Data

XmR charts have an advantage over np, p, c, and u charts in that they require fewer and less stringent assumptions. They are also easier to plot and use. This gives XmR charts wide applicability, and many quality control professionals recommend their use almost exclusively over np, p, c, and u charts. Nevertheless, when the assumptions of the underlying statistical model are met, the more specialized np, p, c, and u charts can give better bounds for control limits. Hence, in the right situations, they offer definite advantages.

The control charts in Figure 5-20 are categorized according to the underlying distributional model they rely on and the nature of the area of opportunity as follows:

- An np chart is used when the count data are binomially distributed and all samples have equal areas of opportunity. These conditions can occur in manufacturing settings, for example, when there is 100% inspection of lots of size n (n constant), and the number of defective units in each lot is recorded.
- A p chart is used in lieu of an np chart when the data are binomially distributed but the areas of opportunity vary from sample to sample. A p chart could be appropriate in the inspection example above if the lot size n were to change from lot to lot.

  In software settings, for instance, it might be possible to use p charts to study coding practices, where the use of a practice is characterized by the percent of code in a module that contains a given construct (a comment, "uses" clause, or "with" clause, for instance). To do this, though, we should first ascertain that the conditions for a binomial model are satisfied, namely that the event counted is matched to individual lines of code in a way such that each line can be treated as an independently drawn sample from a population with a constant probability of event occurrence from line to line—like tossing an unbalanced coin. Thus the inclusion of headers in comment

counts would likely invalidate the use of a binomial model, since headers tend to cluster at the start of modules. This illustrates the kind of care and caution that are needed when using the traditional charts for attributes data.

- A c chart is used when the count data are samples from a Poisson distribution, and the samples all have the same sized areas of opportunity. Under the right situations, c charts can be appropriate charts to use when tracking the number of defects found in lengths, areas, or volumes of fixed (constant) size. As when using a binomial distribution, the justification for assuming a Poisson process should always be examined carefully, and the validity of the assumptions should be verified by empirical evidence wherever possible.
- A u chart is used in place of a c chart when the count data are samples from a Poisson distribution, and the areas of opportunity are not constant. Here the counts are divided by the respective areas of opportunity to convert them to rates. Defects per thousand lines of code or defects per function point are possible examples.
- An XmR chart can be used in any of the situations above. It can also be used when neither a Poisson nor a binomial model fits the underlying phenomena. Thus, XmR charts are especially useful when little is known about the underlying distribution or when the justification for assuming a binomial or Poisson process is questionable.

The first four charts (np, p, c, and u charts) are the traditional control charts used with attributes data. Each assumes that variation (as measured by sigma) is a function of the mean of the underlying distribution. This suggests that caution is in order. Whenever a binomial or Poisson model is truly appropriate, the opportunities for process improvement will be somewhat constrained. The linking between sigma and the mean says that you cannot reduce variability without changing the center line, nor can you move the center line without affecting the variability. If these constraints on process improvement do not make sense, you should not use the traditional charts.

Because attributes data are individual values, XmR charts are almost always a reasonable choice. The exception occurs when the events are so rare that the counts are small and values of zero are common. Then the discreteness of the counts can affect the reliability of the control limits in XmR charts. Still, whenever the average of the counts exceeds 1.00, XmR charts offer a feasible alternative to the traditional control charts described above. And when the average count exceeds 2.00, the discreteness of the counts will have only negligible affects on the effectiveness of the control limits [Wheeler 95].

U Charts

Of the control charts that rely on distributional models, the u chart seems to have the greatest prospects for use in software settings. A u chart is more flexible than a c chart

because the normalizations (conversions to rates) that it employs enable it to be used when the areas of opportunity are not constant.

U charts and c charts both assume that the events that are counted follow a Poisson process. One situation where a Poisson model might be appropriate occurs when counting the numbers of defects found in modules during inspection or testing. If the areas of opportunity for observing defects are the same across all modules or tested components, c charts may be appropriate charts to use. However, if the areas of opportunity are not constant (they rarely are), then the raw counts must be converted to rates by dividing each count by the size of its area of opportunity.

When areas of opportunity are appropriately measured and a Poisson model applies, u charts are the tool of choice. Defects per thousand lines of source code, defects per function point, and system failures per day in steady-state operation are all examples of attributes data that are candidates for u charts.[20] Defects per module and defects per test, on the other hand, are unlikely candidates for u charts, c charts, or any other charts for that matter. These ratios are not based on equal areas of opportunity, and there is no reason to expect them to be constant across all modules or tests when the process is in statistical control.

By virtue of their dependence on Poisson distributions, both u charts and c charts must satisfy the following conditions [Wheeler 92]:

- They must be based on counts of discrete events.
- The discrete events must occur within some well-defined, finite region of space, time, or product.
- The events must occur independently of each other.
- The events must be rare, relative to the opportunity for their occurrence.

We suggest adding two more tests to Wheeler's list:

- The measure used to describe the size of the area of opportunity should be such that the expected (average) number of events observed will be proportional to the area of opportunity.
- No other factor that varies from one examined entity to another materially affects the number of events observed.

---

[20] Although u charts may be appropriate for studying software defect densities, we are not aware of any empirical studies that have validated the use of Poisson models for these situations. We can conceive of situations, such as variations in the complexity of internal logic or in the ratios of executable to nonexecutable statements, where simply dividing by module size provides inadequate normalization to account for unequal areas of opportunity. For example, if modules are deliberately made small when the tasks that they perform are inherently difficult to design and program, then simple defect densities are unlikely to follow the same Poisson process across all modules.

One way to test whether or not a Poisson model might be appropriate is this: If you can count the nonconformities but it is impossible to count the conformities, you may have a Poisson situation. (The conditions that were just listed must apply as well.)

When the opportunities for observing the event are not constant, as when differently sized portions of code are examined or tested for defects, the counts must be converted into rates—such as defects per thousand lines of code or defects per thousand source statements—before they can be compared. The rates are computed by dividing each count (c_i) by its area of opportunity (a_i). The rate that results is denoted by the symbol u_i. Thus,

$$u_i = \frac{c_i}{a_i}$$

Once values for u_i have been calculated, they can be plotted as a running record on a chart, as shown in Figure 5-21.

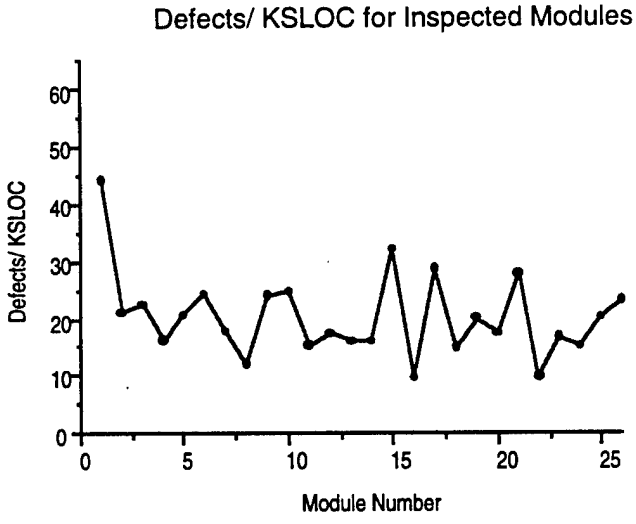


Figure 5-21: Time-Sequenced Plot of Code Inspection Results

The center line and control limits for u charts are obtained by finding $\bar{u}$, the average rate over all the areas of opportunity, and using these formulas:

$$\bar{u} = \frac{\sum u_i}{\sum a_i}$$

$$CL_u = \bar{u}$$

$$UCL_u = \bar{u} + 3\sqrt{\frac{\bar{u}}{a_i}} \qquad LCL_u = \bar{u} - 3\sqrt{\frac{\bar{u}}{a_i}}$$

The presence of the area of opportunity a_i in the formulas for the control limits for u charts means that the limits will be different for each different area of opportunity. This can be disconcerting when the results are presented to people who are not familiar with these kinds of charts or the implications of a Poisson distribution.

The following example illustrates the use of u charts to investigate the stability of a code inspection process.

Example

Figure 5-22 shows a sequence of data obtained from inspecting 26 software modules. The data include the module identifier (i.e., its position in the sequence), the number of defects found that are attributable to coding errors, the size of the module, and the density of defects per thousand lines of code. There are more than 100 other modules in various stages of development that are yet to be inspected, and the questions are

- Are the inspection teams performing consistently?
- Are the results (the number of defects found by the inspection teams) within the historical ranges that we used when planning our resource and time allocations?

To address these questions, which implicitly deal with relations of cause to effect and predicting future performance, we must include the defect insertion (code generation) subprocess within the envelope of the process we are examining. We do this because, based on the data alone, there is no basis for attributing the number of defects found solely to the performance of the inspection teams. The variations we see could easily be caused by differences in the quality of code produced for different modules. When defects are scarce, inspection teams may have a hard time finding them. If so, it would be incorrect to attribute low discovery rates solely to the performance of the inspection teams.

Module Number	Number of Defects	Module Size (SLOC)	Defects per KSLOC
1	19	430	44.2
2	8	380	21.1
3	3	134	22.4
4	6	369	16.3
5	9	436	20.6
6	4	165	24.2
7	2	112	17.9
8	4	329	12.2
9	12	500	24.0
10	8	324	24.7
11	6	391	15.3
12	6	346	17.3
13	2	125	16.0
14	8	503	15.9
15	8	250	32.0
16	3	312	9.6
17	12	419	28.6
18	6	403	14.9
19	3	150	20.0
20	6	344	17.4
21	11	396	27.8
22	2	204	9.8
23	8	478	16.7
24	2	132	15.2
25	5	249	20.1
26	10	435	23.0
Total	173	8316	—

Figure 5-22: Data from Code Inspections

The question then is whether the overall system is performing consistently over time and across all modules with respect to the combined operation of the subprocesses that insert and discover defects. If it is not, early detection of process instability could point us toward assignable causes and let us make corrections to the coding and inspection processes that would make the remaining work predictable.

When we plot the data from the first 26 modules in the sequence in which the inspections were performed, we get the results shown previously in Figure 5-21. To determine whether or not the inspection process is stable, we must now compute appropriate values for the control limits. To do this, the value for the center line $\bar{u}$ is first calculated by dividing the total number of defects found by the total amount of software inspected.

Thus,

$$\bar{u} = \frac{173}{8316} = 20.8 \text{ defects per KSLOC}$$

The upper and lower control limits are then calculated for each point individually, using the formulas stated previously. The calculation of the upper control limit for the first point is illustrated below.

$$a_1 = 0.430 \text{ KSLOC}$$

$$UCL_{u_1} = \bar{u} + 3\sqrt{\frac{\bar{u}}{a_1}} = 20.8 + 3\sqrt{\frac{20.6}{0.430}} = 20.8 + 20.9 = 41.7$$

Corresponding computations would then be made for the lower control limit, and the two computations would be repeated for each point, using the value of a_i appropriate to the point ($a_2 = 0.380$, $a_3 = 0.134$, and so forth).

The completed u chart is shown in Figure 5-23.

The first point in Figure 5-23 falls above its upper control limit. This suggests that the process might not have been in control when it started and that an assignable cause might be found. The reason for the out-of-control indication should be investigated. If an assignable cause can be found and if the process has

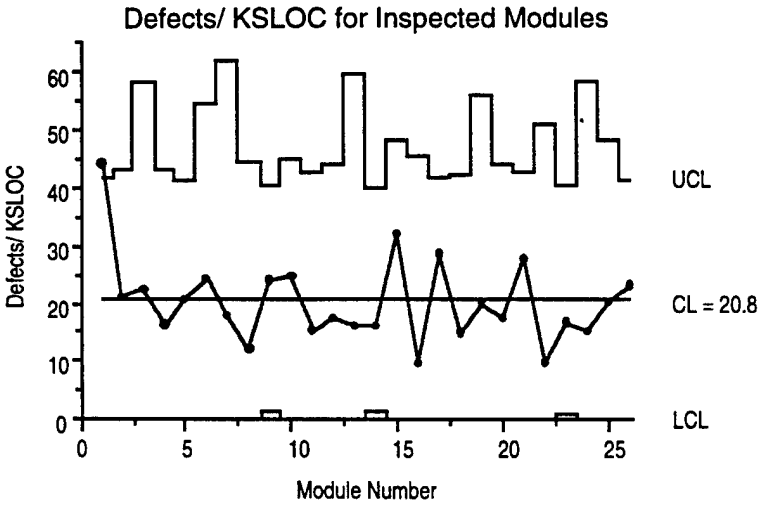


Figure 5-23: U Chart for a Module Inspection Process

changed (or been changed) so that the situation will not recur in the future, we can remove all values that were affected by the assignable cause and recalculate the center line and control limits. This will give us a new center line and control limits that will more accurately characterize the performance of the process as it currently exists.

The results of recalculating the control limits when the first point is removed are shown in Figure 5-24. All individual values are now within the limits, and the average defect rate has been reduced to 19.5 defects per KSLOC. The lowering of the center line occurred as a result of omitting the point that was responsible for the initial instability. Omitting this point is a legitimate step if the situation that caused the out-of-limits point can never recur. From here on, improvements in the performance of the process will require changes to the process itself.

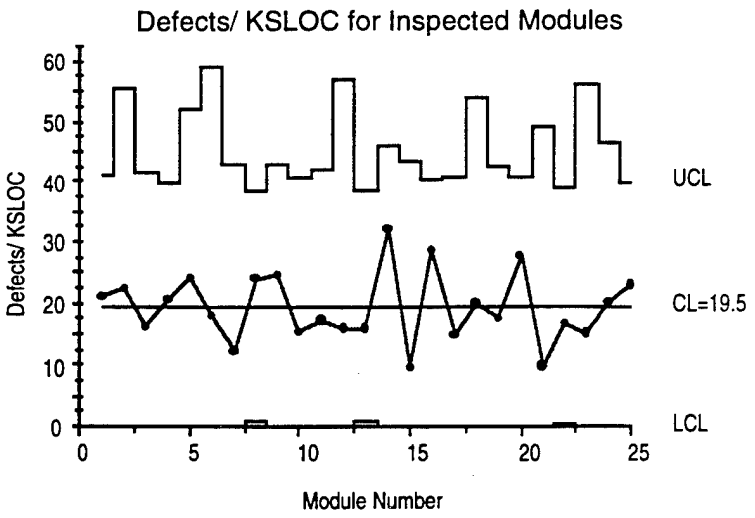


Figure 5-24: U Chart with Assignable Cause Removed

It may be, though, that the example we have just shown is not really a good illustration of the effective *use* of control charts. Since the point that we eliminated represents the discovery of a higher than usual number of defects, the inspection process used for module 1 may have been decidedly more effective than the process used for all modules that followed. If the investigation of assignable causes finds this to be the case, we should seriously consider reinstalling the initial process, throwing out the last 25 data points, and starting over again! This is one way that control charts can provoke process improvement.

On the other hand, if the reason for the high defect discovery rate in the first module lies in the module itself or in the process that produced it, the example may be valid as given. It all depends on whether or not we can find the cause of the defect-rich module and prevent future recurrences. (No one ever said that investigating and correcting assignable causes would be automatic or easy!)

Figures 5-23 and 5-24 illustrate the variable control limits that are associated with using distributional models whose mean and standard deviation are determined by a single parameter. These variable control limits can make the chart seem difficult to analyze and interpret. This problem can be partially avoided when variations in the areas of opportunity are small, say less than 20%. Then control limits can be approximated by constant limits that are based on the average area of opportunity. Exact control limits can always be calculated later, if any points fall near the approximate limits.

Z Charts

Another technique for avoiding variable control limits, and one that works for both large and small variations, is to convert the u chart into a Z chart. Here the individual rates u_i and the average rate $\bar{u}$ are used to compute values that are scaled in terms of sigma units. To construct a Z chart, a sigma value is computed for each point from the equation

$$\sigma_i = \sqrt{\frac{\bar{u}}{a_i}}$$

The *sigma*-equivalent value for each rate is then

$$Z_i = \frac{u_i - \bar{u}}{\sigma_i}$$

These values can be plotted just as on any other control chart. The resultant Z chart for the data in Figure 5-22 is shown in Figure 5-25. This plot has the same general shape as that of Figure 5-23, except that the variation from the center line is now expressed in sigma units. This makes it easier to see nonrandom patterns and test for conditions of instability, as illustrated in previous examples.

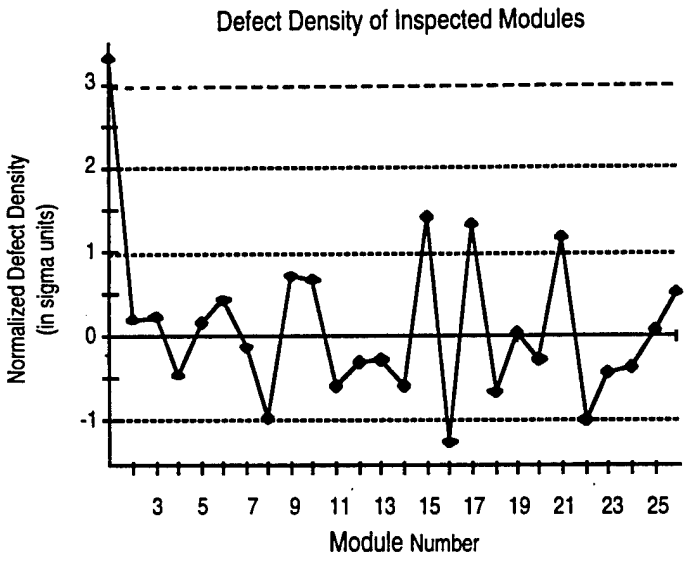


Figure 5-25: An Example Z Chart

XmR Charts for Attributes Data

A third and perhaps the simplest technique for avoiding variable control limits when studying attributes data is to plot the data on individuals and moving range charts (XmR charts). When you do this, you will be abandoning the Poisson (or binomial) model and assuming instead that the standard deviation (sigma) of the phenomena you are studying is constant across all observations when the process is in statistical control. Be cautious though. If you are plotting rates and a Poisson or binomial model truly applies, this will *not* be the correct thing to do, unless the areas of opportunity are all nearly the same size. When the areas of opportunity vary by more than 20% or so, you must examine your assumptions carefully and test them for validity. For example, XmR charts may not be the best tools to use for studying defect densities, if your modules differ widely in size and you see or have reason to believe that the standard deviation of defect density decreases as the size of a module increases (as happens in Poisson processes).

Let us return to the u-chart example and suppose for the moment that we have no theory or evidence that suggests that the standard deviation of defect density decreases as module size increases. Then XmR charts may make more sense than u charts, and they are much easier to plot. The procedure for constructing XmR charts for attributes data is exactly the same as for variables data. This procedure was illustrated earlier when preparing Figures 5-15 and 5-18. If we use that procedure to analyze the data in Figure 5-22, we get the results shown in Figures 5-26 and 5-27.

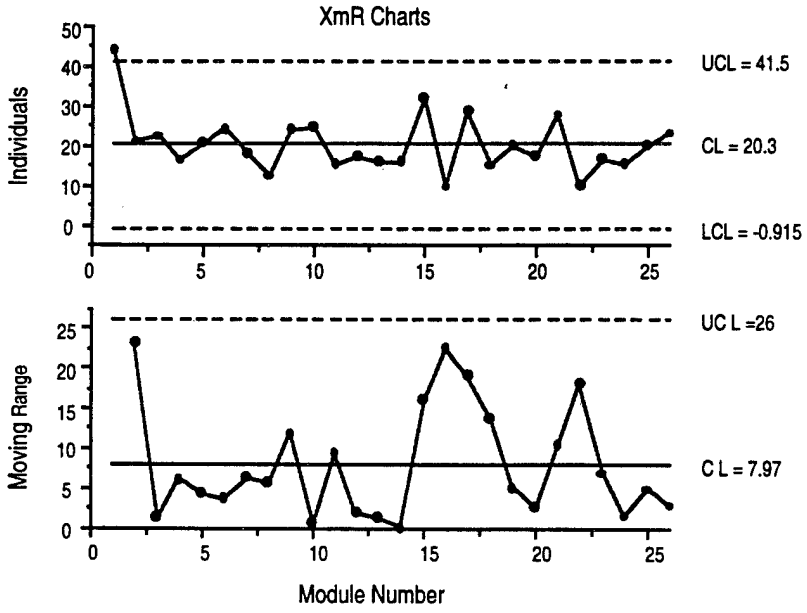


Figure 5-26: XmR Charts for a Module Inspection Process

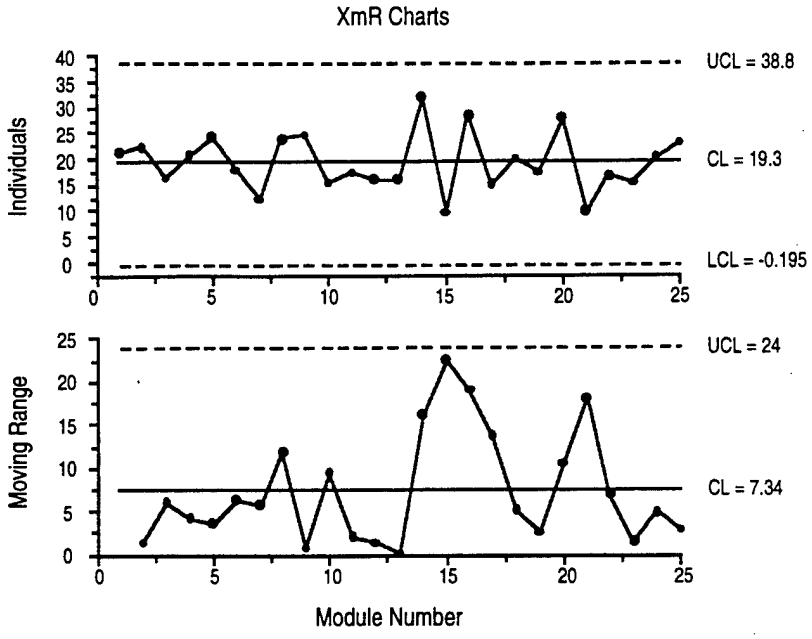


Figure 5-27: XmR Charts with Assignable Cause Removed

Notice that in this example we arrive at the same conclusions that we reached when using u charts and Z charts alone: with the first observation included, the charts signal an out-of-control condition at the initial point; with the first observation removed, the data appear to describe a stable process.

So, which charts should you use? In practice, it is likely that none of the traditional charts will be exactly right. The "rightness" of a control chart all depends on the validity of the underlying probability model—and you can't escape having a model. A probability model is always there, even if you don't explicitly articulate it. Sometimes the model has a specific distributional form, such as binomial or Poisson (there is seldom need to assume a normal distribution). At other times the model simply assumes that sigma is independent of the mean, or that the distribution is symmetric. In most (but not all) situations, the model requires that processes in statistical control produce observations that are independent and identically distributed random variables. Keep in mind, too, that all models are approximations. Real-world data seldom, if ever, follow any theoretical distribution.

If you are uncertain as to the model that applies, it can make sense to use more than one set of charts, as we have just done for the data in Figure 5-22. For instance, if you think that you may have a Poisson situation but are not sure that all conditions for a Poisson process are present, plotting both a u chart (or c chart) and the corresponding XmR charts should bracket the situation. If both charts point to the same conclusions, you are unlikely to be led astray. If the conclusions differ, then investigations of either the assumptions or the events are in order. If you choose to investigate the events, the worst that can happen is that you will expend resources examining an in-control process.

5.6 Evaluating Process Capability

> *There are two common practices in estimating capability which have no valid statistical foundation. Engineers should avoid using either of them in arriving at estimates or setting standards. The things to avoid are the following:*
>
> *I. Do not attempt to use a distribution without a control chart.*
>
> *II. Do not attempt to use the average of past data without a control chart.*
>
> *Statistical Quality Control Handbook*
> *[Western Electric 58]*

Whenever we propose to produce products to meet specifications or to meet deadlines at a specified cost with acceptable quality, we find ourselves relying, at least implicitly, on the concept of *process capability*. This concept applies only to stable processes. Before a process can be said to have a defined capability, it must display a reasonable degree of statistical control. This means that evaluations of process capability must proceed in two stages:

1. The process must be brought into a state of statistical control for a period of time sufficient to detect any unusual behavior.
2. The stable performance of the process must then be compared to the specifications that have to be satisfied to meet business or customer requirements.

Attempting to assess the capability of a process that is not in statistical control is fruitless. Evaluations of capability invariably get interpreted as predictions of future performance, and it is unrealistic to expect records of past performance to be reliable predictors of the future unless you know that the underlying system of chance causes is stable.

On the other hand, when a process is in statistical control with respect to a given set of attributes, we have a valid basis for predicting, within limits, how the process will perform in the future. As long as the process continues to remain in the same state of statistical control, virtually all measurements of those attributes will fall within their natural process limits.

Capability Histograms

The simplest and easiest way to assess the performance of a stable process with respect to a given attribute is to plot a histogram of the individual values that have been observed during a period of stability. The natural process limits (i.e., the control limits for individual values) can then be used to characterize the performance. These limits can be computed

from the grand average and average range of the measured values by using the following formula:

$$\textit{Natural Process Limits} = \overline{\overline{X}} \pm 3\,sigma_X = \overline{\overline{X}} \pm 3\frac{\overline{R}}{d_2}$$

The ranges that are averaged can be either those of the subgroups used to construct an X-bar chart or the moving ranges used to plot an XmR chart. When two-point moving ranges associated with XmR charts are used, d_2 will equal 1.128, and the equation reduces to

$$\textit{Natural Process Limits} = \overline{\overline{X}} \pm 2.660\overline{R}$$

Figure 5-28 shows a histogram of the 80 individual values recorded in Figure 5-11 (the Homogeonics example). Since our analyses of these data in Figures 5-12 through 5-15 showed no signs of lack of control, we are justified in computing the natural process limits and plotting them on the histogram. These limits, which are the same as the upper and lower limits that would be used on a control chart for individual values, are designated in Figure 5-28 by UCL and LCL. For the Homogeonics data, all 80 of the measured values fall within the range given by the natural process limits.

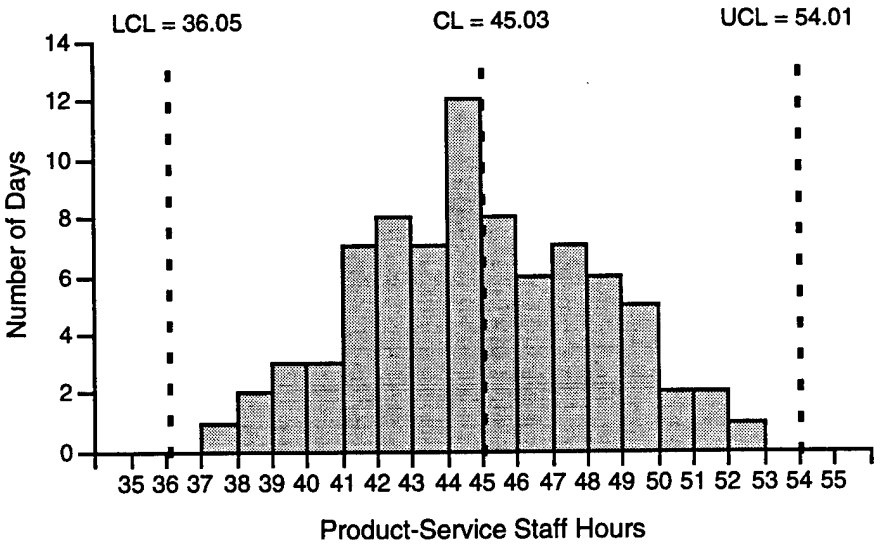


Figure 5-28: A Process Capability Histogram

The natural process limits in Figure 5-28 represent the voice of the process. They reflect what the process can be expected to do as long as it continues to operate as it has over the past 80 days. A stable process will continue to produce results that are almost always within its natural process limits until some event occurs that causes the process or its inputs to change.

The second stage of a capability analysis consists of comparing the natural process limits to the specifications imposed by business needs or customer requirements. When the natural limits fall entirely within the specification limits, almost every product that is produced will conform to specifications. Such a process is said to be *capable*. If a process is not capable, it will produce nonconforming results at rates that may be economically inefficient.

Thus a simple way to assess whether or not a process is capable is to plot the specification limits on the histogram of measurements obtained from a stable process. The relationship of the natural process limits to the specification limits will portray the capability of the process.

Suppose, for instance, that the software manager at Homogeonics, Inc. requires product-service staff hours per day to fall between a value of 30 at the lower limit (to guarantee that customer requests are logged and acknowledged) and 50 at the upper limit (to ensure that critical skills are available for other activities). If the distribution is expected to be symmetric, the optimal value for the average daily product-service staff hours will be 40, the midpoint of the desired range, as used in his plans. The manager's requirements, which are effectively his specifications, are illustrated in Figure 5-29. (LSL and USL are the lower and upper specification limits, respectively.)

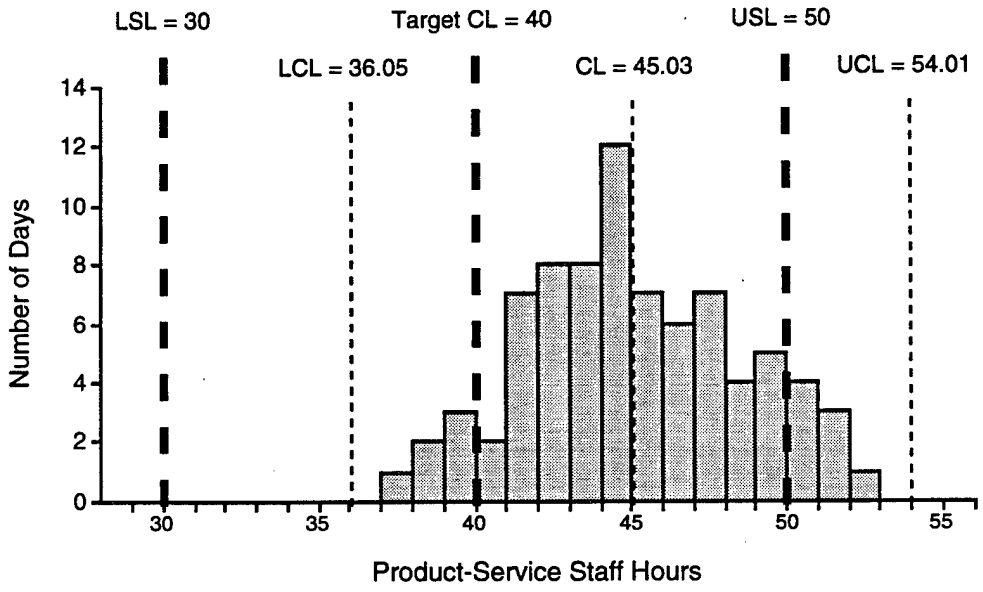


Figure 5-29: Process Capability Histogram with Specification Limits

Fraction Nonconforming

Individual values that fall outside the specification limits are instances of nonconformance to specifications. The fraction of the total observed outcomes that are nonconforming in a stable process gives an unbiased estimate for the fraction of nonconforming outcomes

associated with the process. The statistical term for this is *fraction nonconforming.* In the Homogeonics example, 5 out of the 80 days had nonconforming results (the data in Figure 5-11 show 5 values above 50 and none below 30). This suggests that about 6.25% of the days in the underlying population experience service staff hours that exceed specifications.

Some caution is in order here. Although the observed fraction nonconforming is an unbiased estimate of the true fraction nonconforming in a process, it is seldom a precise estimate. Unless you have many more than 80 data points, the uncertainties will be much larger than you might expect. For instance, when n is large (greater than 30, say), the observed fraction nonconforming f will be approximately normally distributed. We can then compute (approximate) 95% confidence intervals for the true fraction nonconforming p as follows:

$$p = f \pm 1.96\sqrt{\frac{f(1-f)}{n}}$$

For the Homogeonics data, this yields

$$p = .0625 \pm 1.96\sqrt{\frac{.0625(.9375)}{80}} = .0625 \pm .0530,$$

an interval that ranges from 0.0095 to 0.1155 (0.95% to 11.55%). These numbers are not precise, however, since the normal approximation to the binomial distribution becomes less reliable as f approaches 0 or 1. When the observed fraction nonconforming is close to these limits, you may prefer exact methods like those described in Section 7-3 of the National Bureau of Standards handbook *Experimental Statistics* or in Hahn and Meeker's book *Statistical Intervals* [Natrella 66, Hahn 91]. For the Homogeonics data, the exact method yields a 95% confidence interval for the true fraction nonconforming that ranges from 2.5% to 14.5% (interpolated approximately from Natrella's graphs).

Although estimates based on empirical counts of the fraction nonconforming make few assumptions, they are not necessarily the best estimates for the true fraction nonconforming. The reason is that counting the number of items that fall outside specification limits uses only a portion of the information contained in the full set of measured values. Sometimes better estimates are possible—if certain conditions are present.

For instance, if you have reason to believe that the variation in a stable process is the sum of several independent small effects and that no individual effect is much larger than the others, you may be justified in assuming that the measurements approximate independent and identically distributed observations from a normal distribution.[21] When independence, identicality, and normality are present, you can use the normal distribution to estimate the fraction nonconforming in the underlying distribution. The procedure and tables for doing this

---

[21] Shewhart and Wheeler both point out that assumptions like this are essentially unverifiable. Caution is warranted when assuming normality, just as when assuming the relevance of any other distribution [Shewhart 39, Wheeler 92].

are given on pages 58–59 and 133 of the Western Electric handbook [Western Electric 58]. In the Homogeonics case, this procedure gives an estimate of 4.7% nonconforming.

Despite their familiarity and intuitive appeal, however, confidence intervals for the fraction nonconforming (and for other parameters of the underlying process) may not be the proper tools to use. What really affects your decisions are not the uncertainties in estimates for the underlying parameters, but uncertainties associated with future outcomes. The probability that the next result will fall outside the specification limits, for example, has direct economic consequences, as does the proportion of future results that will be nonconforming. Confidence intervals do not address these issues. Instead, you should use prediction intervals (if concerned with one or more specific future observations) or statistical tolerance intervals (if concerned with uncertainties in the fraction nonconforming in all future outcomes). The calculations for these intervals are generally no more difficult than those for confidence intervals [Hahn 93]. If you wish to pursue this subject (we encourage you to do so), you can find discussions of prediction and tolerance intervals and methods for computing them described in papers by Hahn, Scheuer, and Vardeman and in Hahn and Meeker's book *Statistical Intervals* [Hahn 70, 91, Scheuer 90, Vardeman 92].

Capability Indices

In lieu of the graphical summary of capability that a histogram provides, many people compute numerical summaries. Values with names like C_p and C_{pk} are often used in an attempt to reduce measures of capability to a single index. Although these indices are often used alone, they are more effective when combined with a histogram. Deming explains why:

> *The fallacy of an index of dispersion, widely touted by beginners taught by hacks, is now obvious. An index of dispersion has no meaning, because the loss entailed depends far more on the position of the centre of the distribution of production than on its standard deviation.*
>
> *W. Edwards Deming, 1993*

Our advice is to stick with graphical representations. They are not only more informative, but also easier to explain than excessively condensed indices. If someone insists on having a value for C_p or C_{pk}, provide it, but provide the graphical summary as well. You will both be better served.

Specification Tolerances

When one (or both) of the natural process limits falls outside the specification limits, as happens in Figure 5-29, the process is likely to produce nonconforming results more frequently than desired, even when it is in statistical control. For example, in Figure 5-29 the process is stable, but 6.25% of the results have fallen outside the specification limits.

Processes with a large number of nonconforming results can be *stable*, but they are not *capable*. To reduce the amount of nonconforming product or the frequency of nonconformance to other process standards, action must be taken to change the process. The changes must have one or more of the following objectives: reduce process variation, reset the process average, or relax the specifications.

You can test whether or not specification limits are wide enough to allow for natural process variation by calculating the specification tolerance—the distance between the specification limits—as follows:[22]

$$\text{Specification Tolerance} = \text{Upper Spec} - \text{Lower Spec}$$

$$\text{Specification Tolerance (in sigma units)} = \frac{\text{Specification Tolerance}}{sigma_X}$$

$$\text{where } sigma_X = \frac{\overline{R}}{d_2}$$

When the tolerance exceeds six sigma units and the process is centered within the specification limits, the specification leaves sufficient room for process variation without producing large amounts of nonconforming product. When the specification tolerance is less than six sigma units, extreme values will frequently exceed the specification limits, regardless of the centering.

One reason a process may not be capable is that it may not be sufficiently centered. That is, the average of its measured values may not fall at an appropriate point within the specification limits. For symmetric distributions, the way to remedy this problem is to change the process so that the process average is brought closer to a value midway between the limits. We can determine how much adjustment is needed to center the process average by calculating the distance to the nearest specification (DNS).

First, some background: the distance in sigma units between the process average and the upper and lower specification limits (USL and LSL) is given by:

$$Z_U = \frac{USL - Process\ Average}{sigma_X}$$

$$Z_L = \frac{Process\ Average - LSL}{sigma_X}$$

If the process average is within the specification limits, the values of Z_U and Z_L will both be positive. If the process average is not within the specification limits, one of the Z values will

---

[22]Specification tolerances (sometimes called engineering tolerances) are related to statistical tolerance intervals only when the upper and lower specification limits have been chosen based on an analysis of statistical tolerance intervals.

be negative. To hold the number of nonconforming items to an economical value, it is desirable that both Z values be positive and greater than three. When both values are positive, the smaller of the two is called the distance to the nearest specification (DNS).

Figure 5-29 shows a process limit that exceeds the specification limits, so we can use the example to illustrate the procedures just described. To satisfy the specifications, the tolerance must exceed six sigma units. To check this, we compute

$$\text{Specification Tolerance (sigma units)} = \frac{USL - LSL}{sigma_X} = \frac{50 - 30}{\overline{mR} / d_2}$$

$$= \frac{20}{(3.38 / 1.128)} = 6.67$$

Thus, the process is currently meeting this requirement. The distance to the nearest specification, however, is

$$DNS = \frac{USL - \overline{\overline{X}}}{sigma_X} = \frac{50 - 45.03}{2.996} = 1.68$$

This is well below the value of three needed to avoid an excessive number of nonconforming results.

To comply with the specifications, Mr. Smith (the manager at Homogeonics, inc.) will have to change the process so that the average number of product-service hours per day decreases. If the variability is symmetric about the average, the optimal value for the new average will be the point midway between the specification limits, coinciding with his original planning target of 40. If this change can be made, the DNS will become

$$DNS = \frac{USL - \overline{\overline{X}}}{sigma_X} = \frac{50 - 40}{2.996} = 3.34$$

The tactics that Mr. Smith might use to shift the mean include improving a number of attributes or subprocesses. Product quality, user documentation, customer training, the handling of service requests, and the process that assigns staff to service requests are possible examples. Of course, any changes that he introduces will create a new process. The performance of the new process should then be monitored and plotted on run charts, and new control limits should be constructed as soon as sufficient data are available. Once the process appears to be stable, its capability can be re-evaluated.

The advantage of using the specified tolerance and the DNS to characterize the capability of a stable process is that the two measures focus attention on the areas that need improvement. Instead of using artificial index numbers or simple values for percent defective or percent nonconforming, the DNS relates directly to the data in ways that make it easy to understand just how the process is performing relative to the customer's specifications.

A Procedure for Assessing Process Capability

The flowchart in Figure 5-30 gives an orderly procedure for assessing process capability [Wheeler 92].

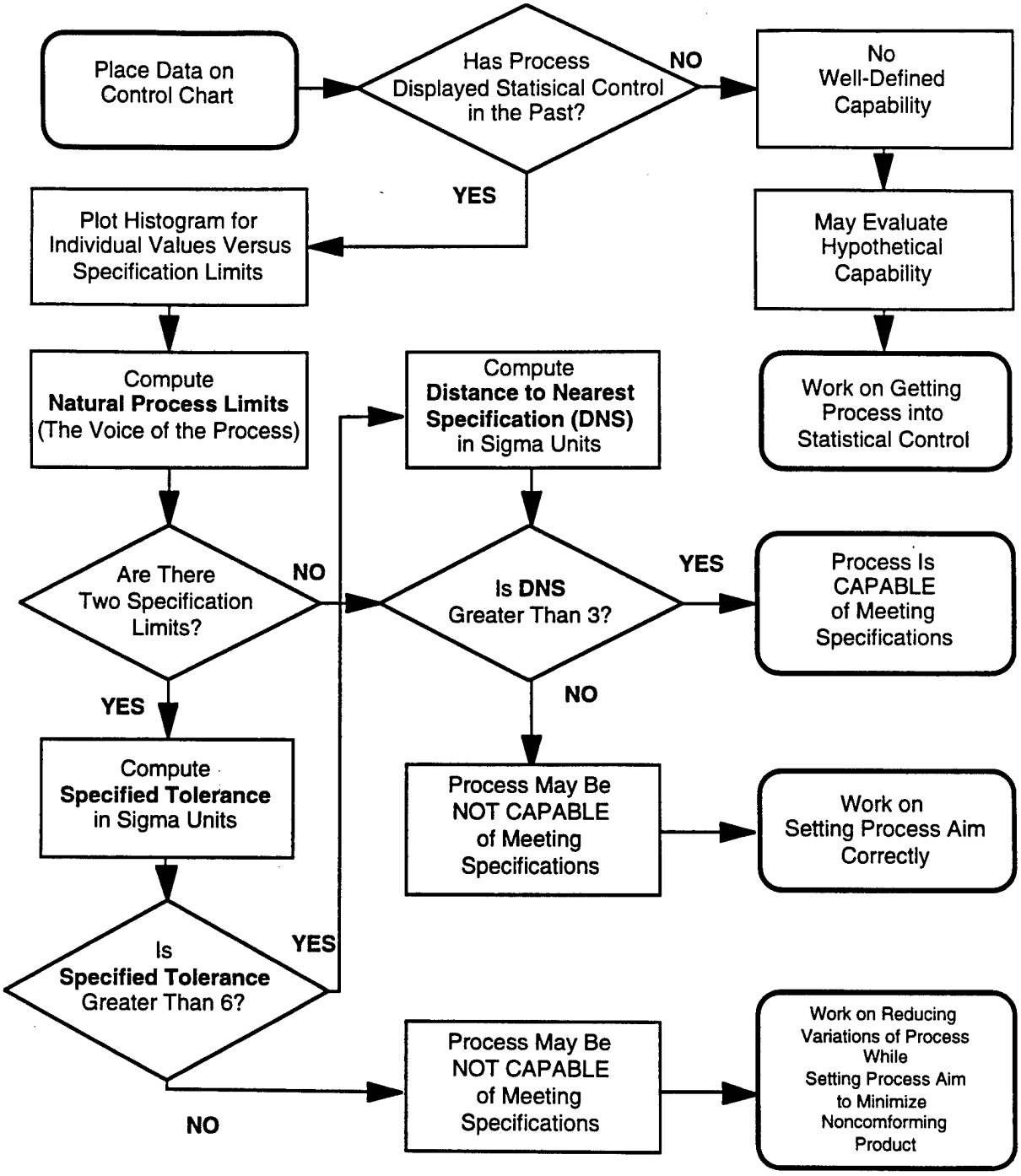


Figure 5-30: Assessing the Capability of a Stable Process

6 Applying Measures to Process Management—Part 3: Acting on the Results

> *Discovering that a process is out of control is not a terrible event. It should not be hidden from supervisors, managers, auditors, quality control experts, or, most important, customers. In a sense, it is an event that should be celebrated because it gives the process owner an opportunity to improve the process.*
>
> *Robert Hoyer & Wayne Ellis, 1996*

We are now at the third step in applying measurements to process management and improvement. The objective here is to translate what you have learned from analyzing stability and capability into actions that will improve the efficiency of your processes and the quality of your products. Figure 6-1 highlights the focus of the chapter.

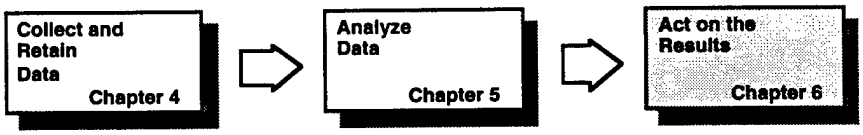


Figure 6-1: The Focus: Act on the Results

Sections 6.1 through 6.3 point out important aspects of investigations aimed at achieving process stability and improved levels of performance. Section 6.4 then gives brief illustrations of some of the more useful tools that are available to help you with your investigations. We will not teach you how to use the tools, but we will provide you with insights regarding their use that you may not have encountered before. Section 6.5 closes the chapter by listing methods and technologies that organizations have found effective for improving software products and processes.

6.1 General Principles

The Additional Roles of Measurement

When process measurements are analyzed as in Chapter 5, the results point to one of three investigative directions:

- **Stability**. If the process is not stable, the proper action is to identify the assignable causes of instability and take steps to prevent the causes from recurring. Decomposing a process into its subprocesses and examining the

subprocesses for stability is often a productive approach for pinpointing the roots of assignable causes.

- **Capability**. If the process is stable but not capable (not meeting customer needs), the proper action is to identify, design, and implement changes that will make the process capable. Keep in mind that changing a process transforms it into a new process, and the new process must be brought under control and made stable before its capability can be assessed.
- **Improvement**. If the process is both stable and capable, the proper action is to seek ways to continually improve the process, so that variability is reduced and quality, cost, and cycle time are improved. Brainstorming, designed experiments, multivariate analyses, and investigations of subprocesses can all provide helpful insights into ways for improving process performance. Once more, any changes to a process will transform the process into a new process whose stability must be established before you can rely on predictions of future performance.

Figure 6-2 shows the how these actions relate operationally to the business goals and strategies of an organization. The actions that should follow stability and capability measurement are highlighted by the shaded box.

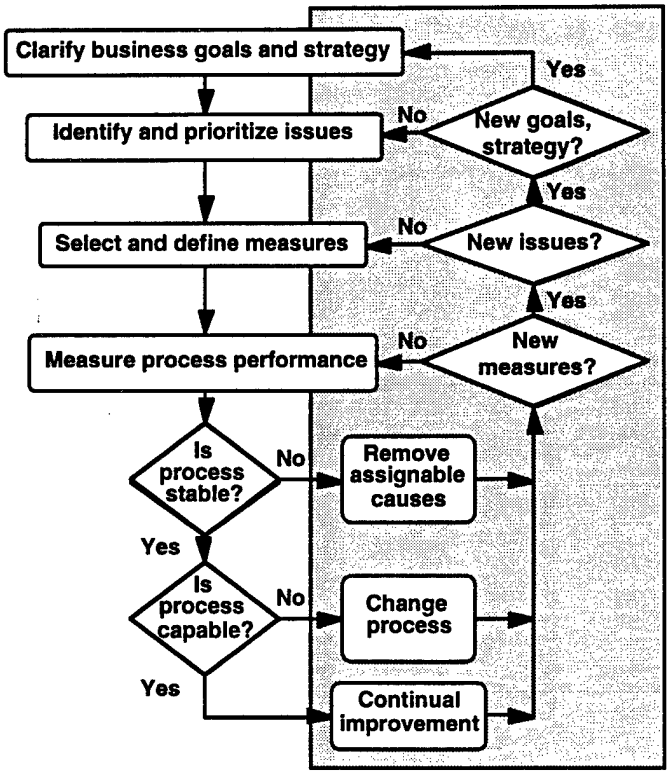


Figure 6-2: The Actions that Follow Evaluations of Process Performance

Although Figure 6-2 gives a good overview of the logical process described in this guidebook, there is more to measuring for process management and improvement than the figure suggests. For example, there is

1. the role that measurement (and perhaps designed experiments) plays in finding, confirming, and eliminating root causes of instabilities and inadequate performance. (What is causing the problems we see?)
2. the role that measurement (and perhaps designed experiments) plays in identifying cause-effect relationships. (Which factors should we change when we wish to improve?)
3. the role that measurement plays in quantifying cause-effect relationships. (How large a change should we make?)
4. the role that measurement plays in estimating costs and benefits associated with present levels of performance and proposed improvements. (What are the costs and benefits now? What will they be after the changes? What will it cost to make the changes? How long will it take?)
5. the role that measurement plays in obtaining descriptive information that can guide interpretations and actions. (What are the other characteristics of the product, process, system, environment, resources, and customer that we are addressing?)

We have talked about the fifth role (gathering contextual information) in Chapters 2 and 4. The purpose of this chapter is to address roles 1–4 and provide pointers to tools that can help you get the other information you will need to turn the insights you have gained into actions that work.

The Need for Additional Information

> *Statistical analysis of data that just happens to be available is* ***very*** *risky.*
>
> *Thomas Pyzdek, 1992*

Once you find that a process is either unstable or performing at an unacceptable level, you will almost always need additional information to help you find the assignable causes and devise remedial actions. Some of this information may well exist now, somewhere within the organization. You should strive to find it and pull it together. Other kinds of information may not have been assembled before. Getting this information may mean measuring additional attributes of the product, process, or environment.

There are three reasons for gathering additional data:

1. to support or invalidate hypotheses about possible causes of instability
2. to identify, verify, or quantify cause-effect relationships, so that the right amounts of the right changes can be made
3. to estimate (or confirm) the business value of the actions that are proposed (or taken) to improve the process

There is also the pragmatic reason that, whatever action you propose, you will have to convince others to assist you in funding and implementing the changes. Without factual evidence to support the rationale for proposed actions, people will have every right to challenge your proposals.

Perceptive readers will observe that the discussion above exposes the recursive nature of measurement—at least as it applies to process management and improvement. Discovering that a problem exists is only the start of a journey. The next questions are "Why?", "What can we do about it?", and "How much is it worth?" Obtaining reliable data to identify and justify effective actions means returning to the same logical process that we have been illustrating for detecting the existence of problems:

select → define → collect → analyze → act

The analytic methods that you will use to identify root causes and solutions, though, may be different from the ones you used for investigating process stability and capability. Section 6.4 will illustrate just a few. You can find others discussed in almost any treatise on quality improvement or creative problem solving. Two often overlooked tools that we call to your attention are design of experiments and multivariate analysis. If software engineering is to continue to mature along the lines of other scientifically based disciplines, these tools are likely to see increasing use.

When you begin your search for explanatory data, there are general principles that you should recognize. Ott and Schilling offer three [Ott 90]:

Rule 1: Don't expect many people to advance the idea that the problem is their fault. Rather it is the fault of raw materials and components, a wornout machine, or something else beyond their own control. "*It's not my fault!*"

Rule 2: Get some *data* on the problem; do not spend too much time in initial planning. (An exception is when data collection requires a long time or is very expensive; very careful planning is then important.)

Rule 3: *Always graph* your data in some simple way—*always.*

6.2 Establishing Stability

Finding and Correcting Assignable Causes

> *An assignable cause of variation as this term is used in quality control work is one that can be found by experiment without costing more than it is worth to find it.*
>
> *Walter A. Shewhart, 1939*

When you find a process that is not stable, you should try to find assignable causes and take actions to prevent their recurrence. This calls for detective work that is not unlike that of debugging a failing software component. You will not be far off track if you think of the hunt for an assignable cause as the equivalent of the hunt for the cause of a software failure.

As with software, when process outputs vary erratically, one of the best things to do is to try to isolate the problem (or problems). To begin, you may want to assemble a group of people who work within the process to brainstorm possible reasons for the unusual behavior. Ishikawa charts and Pareto diagrams can help to focus the discussions and summarize the results.[23]

You may also find that you want to measure key parameters of some of the inputs to the process and plot run charts (or control charts) to see if anything unusual is happening. Out-of-control inputs, for example, can cause out-of-control outputs. When out-of-control inputs are found, this will push your search for root causes and corrective actions to points earlier in the overall process.

Parallel input streams of supposedly similar items can also be sources of erratic behavior when the inputs have different characteristics. The symptoms often show up as stratifications in control charts of process performance (stratification patterns show many values toward the outer edges of the control chart and relatively few near the center line).

Another strategy for isolating problems is to begin measuring and plotting characteristics of intermediate products, so that you can better identify where in the process erratic behavior is first making its appearance.

Awareness of the environment and what has been happening within the organization can provide valuable clues to the nature of assignable causes. But important as these clues and insights are, the best that they can do is to help you form hypotheses. You will almost always need data (and perhaps experiments) to test the hypotheses and to support or eliminate specific factors as assignable causes. This means that measurement, with all its

---

[23] Ishikawa charts and Pareto diagrams are illustrated in Section 6.4.

subactivities—selecting, defining, collecting, retaining, analyzing, and acting—will once again be needed.

Sometimes assignable causes, once found, can point the way to improving process performance. For example, an unusual series of events or an inadvertent change to the process may cause the process not only to become unstable, but also to improve—at least momentarily—certain aspects of quality. When this happens and the assignable causes can be replicated, it may be profitable to incorporate the causes of the beneficial results as permanent parts of the process. Finding and making changes that will improve process performance is the topic of Section 6.3. For the moment, when instabilities are present, the first order of business is to isolate and eliminate the elements that are making the process unstable.

It is important to recognize that correcting assignable causes is not the same as changing or improving a process. Eliminating assignable causes merely brings a process to a repeatable state. But this is a very important state to achieve. When a process is not stable, it seldom makes sense to introduce changes to improve capability or performance. Without stability as a baseline to start from, you are likely to simply trade one unstable condition for another.

Noncompliance as an Assignable Cause

Compliance means that standards of knowledge and practice exist and are followed. Compliance also means that the process is supported adequately and that the organization is capable of executing the process. When a process component is not performing consistently, lack of compliance to process standards may be the cause of the instabilities you see in process results.

Processes get designed, either implicitly or explicitly, around four kinds of knowledge: a process definition; the known or estimated performance of process components; the existence and effectiveness of support systems; and the anticipated effects of organizational factors such as management support, organizational changes, and personnel policies and actions. Therefore, when investigating compliance (or the lack thereof) as a potential source of process instability, the following aspects of compliance should be examined:

- adherence to the process
- fitness and use of people, tools, technology, and procedures
- fitness and use of support systems
- organizational factors, such as management support, organizational changes, personnel turnover, relocations, downsizing, and so forth

Figure 6-3 lists some of the things to examine when searching for causes of noncompliance.

Compliance Issues	Things to Examine When Seeking Reasons for Noncompliance
Adherence to the process	awareness and understanding of the process existence of explicit standards adequate and effective training appropriate and adequate tools conflicting or excessively aggressive goals or schedules
Fitness and use of people, tools, technology, and procedures	availability of qualified people, tools, and technology experience education training assimilation
Fitness and use of support systems	availability capacity responsiveness reliability
Organizational factors	lack of management support personnel turnover organizational changes relocation downsizing disruptive personnel morale problems

Figure 6-3: Compliance Issues and Potential Sources of Noncompliance

Establishing and maintaining baselines of information about compliance can help in finding assignable causes. Deviations from baselines when executing a process are potential sources of instability. When historical information is not available to serve as a baseline, you may want to begin measuring aspects of compliance to obtain data that shed light on the origins of process instabilities. Baselines of information about process components and activities should be among your considerations when designing a process measurement database.

Many of the issues and methods associated with obtaining and retaining compliance data have been discussed already in Sections 2.3 and 4.2. Reviewing those discussions can help you get started when you sense that instabilities in process results may be caused by deviations from the intended process.

6.3 Improving the Process

*The most important use of a control chart is to **improve** the process.*

Douglas C. Montgomery, 1996

...most control charts, even if used correctly, are used too late—too far downstream to be of any substantial benefit.

W. Edwards Deming, 1986

Improving process capability requires making changes either to the process or its specifications. Figure 6-4 illustrates the three possibilities:

1. We can reduce variability while keeping the average the same.
2. We can retarget the process by shifting the average.
3. We can revise the specification so that more of the results fall within the specification limits. (As this is a negotiation issue rather than true process improvement, we will not discuss this case further.)

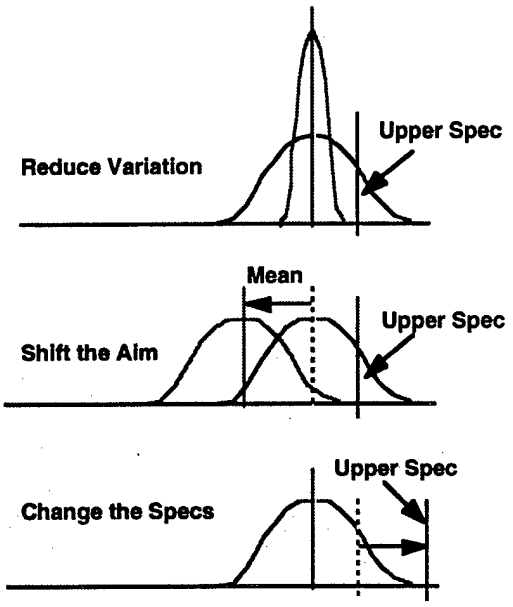


Figure 6-4: Three Ways to Improve Process Capability

Where to Look When Seeking Improvements

It is easy and almost inherent in us to look for and find solutions. It takes discipline to first stop and look for causes.

Tony Burns, in a posting to bit.listserv.quality, January 1997

There are two places to look when seeking ways to improve process performance:

- process activities and subprocesses
- things used within the process that originate outside the process

The knowledge you have of your process (supported by evidence where available) will guide you in directing your attentions inward or outward. It may be that you will want to look in both directions simultaneously.

When looking inward, your strategy will be to decompose (divide and conquer). By breaking a process down into its component activities and subprocesses, you will often find points where intermediate products are produced. The output streams from these processes then become candidates for measurement and analysis, and control charts may once more be appropriate tools—especially if reducing variability is an issue.

When looking outward, just as when eliminating instabilities, the characteristics of the materials, resources, and guidelines you work with can limit what your process can achieve. If materials, resources, or guidelines constrain the performance you seek to improve, you may have to go back to the origins of these entities to find ways to change your level of performance or reduce variation in process results.

Figure 6-5 lists some common process entities that have origins outside the process. These are things that you may want to examine when seeking ways to improve a process. If the stability and variability of attributes associated with these entities can significantly affect the operation of your process, you will want to consider implementing measures that help quantify, control, and improve those attributes. This may easily lead you back to issues like the ones addressed in earlier chapters of this guidebook, but with attention now directed to other parts of the overall software process.

Entities Originating Outside a Process Whose Attributes Can Affect Process Performance	
products and by-products from other processes	guidelines and directions
resources	• policies
• people	• procedures
• facilities	• goals
• tools	• constraints
• raw materials	• rules
• energy	• laws
• money	• regulations
• time	• training
	• instructions

Figure 6-5: Inputs that Affect Process Performance

As one example of the importance of looking upstream, consider a system testing process. The best way to reduce the time, rework, and cost of system testing may lie not in the testing process itself, but in the quality of the incoming artifacts it tests. Improving the economics of testing may not be possible without improving key parts of the upstream

processes—the activities where defects are inserted and the points where prevention, detection, and removal activities can be implemented or made more effective.

When examining the performance of upstream processes, you may find that they in turn are affected by elements that originate from outside. Dictated schedules (time) and tight budgets (money) are just two examples. Restrictions on time and money have been known to lead organizations to scrimp on the efforts they allocate to preventing and detecting defects during design and coding. The upstream processes may meet their time and budget goals, but the downstream impacts can easily outweigh the upstream benefits. In one sense, looking outside the testing process is an issue of balancing the performance of the overall software process (i.e., the system). In another sense, if the organization knows that inspections work well, has data to suggest what the optimal levels of inspection might be, and would usually perform at that level, this may be just another instance of process noncompliance.

Effects of Changing a Process: Look Before You Leap

When you propose actions that are intended to improve a process or adjust it to meet specifications, you should always examine the effects that the changes will have on other processes in the stream of work. (The Law of Unintended Consequences is ubiquitous.)

Downstream effects. Changes that reduce variability while keeping the average about the same will usually be welcomed—they often improve the efficiency of downstream activities. Reductions in variability seldom have detrimental effects. Changes that shift the average, on the other hand, are quite likely to affect downstream processes, and the consequences should be examined closely. It will be unwise to make these kinds of changes unless the new level somehow benefits the people and processes that rely on the results of the process you are improving.

Upstream effects. Not all effects flow downstream. Sometimes there are feedback loops where measured qualities of outputs from a process are used to control or specify attributes of its inputs. For example, high rates of defects found in testing can induce increased investment in inspection, thus changing the quality of outputs from earlier parts of the process.

Knowing and quantifying the benefits of reduced variability or new levels of performance can help greatly in gaining support and funding for investigating ways to move the process mean and for implementing changes to the process to reduce variation or reset its level of performance.

After the Change: Examine the Results

After you change a process to stabilize or improve its performance, you should always examine how the results affect the performance and opportunities for improvement in other processes in the stream of work.

Downstream opportunities. Reduced variability in the inputs to downstream activities can create opportunities for improvement there by making it easier for those activities to isolate and reduce their own local causes of variability. This could make control charting useful where it was not before because natural process variability was getting overwhelmed by outside influences. Reduced variability in inputs to downstream activities can also lead to more precise planning, since the downstream processes will now be able to rely on the quality and timing of the inputs they receive.

Upstream opportunities. Reduced variability in outputs whose measured values are used to control or influence upstream activities can affect the dynamic behavior of the overall system. This can create opportunities for improving system performance that are not obvious unless you look for them. For example, improving performance or reducing variability in one part of a system may permit you to use different methods or sensitivities for detecting and signaling changes in process performance. When the inherent variability is small, for instance, you can often detect and react quickly to shifts in the average level of process performance. Moreover, more predictable performance means that sensors can sometimes be reduced to occasional use or relocated, which may provide opportunities to change the points upstream where control is applied.

Other kinds of opportunities can flow upstream as well. For example, when the performance of a downstream process is improved, the high payoff areas for future improvement often shift upstream. Sometimes these kinds of opportunities for improving overall system performance are not possible until the downstream process you are working on is made more consistent, efficient, agile, and responsive.[24]

Conditions for Success

> *The process is always changing.* ***Always.*** *And in most cases the reduction in variation achieved by eliminating special causes is only a small fraction of the total variation. The real payoff is in reducing variation from* ***common causes.*** *But the mindset of classical, enumerative statistics inhibits this activity. The questioning, probing, exploratory approach combined with analytic tools like the control chart encourage you to find the cause of all of the variation. When evaluating the control chart off-line, forget the control limits. Investigate the chart as a whole, examine the patterns. Relate the data to what the team knows about the process. Place the charts end-to-end on a*

[24]The successes of JIT (just in time) methodologies in industry illustrate the kinds of benefits that are sometimes available through interactions among processes.

large table and look for cycles and patterns across several charts.

Brainstorm. Be OH!pen minded. Think!

Thomas Pyzdek, 1992

To succeed in finding ways to improve process performance, just as when searching for assignable causes, knowledge of the process—its activities, intermediate products, people, tools, and work flows—is paramount. It is this knowledge that points you to likely areas to investigate. This is one reason why responsibilities for process control and process improvement cannot be delegated to statisticians and staff scientists. Specialists can help, but responsibilities for improvement and for knowing the subject matter lie with those who own and operate the processes.

Remember that when you introduce improvements, the improvements will change the process. Changes mean instability, and instabilities lead (at least temporarily) to lack of predictability. You will not get the most from further improvement actions until you achieve stability at the new level of performance.

Does this mean that you should be reluctant to introduce changes? Not at all! At least not when you have stable baselines against which to measure the results of your actions. Without stable baselines, though, you may have difficulty determining whether or not your actions have the effects intended. This means that the wisest first step is often to bring the existing process under control. Then, at least, you will know where you are starting from, and you can measure how well the improvements work. Without baselines, you will just be guessing.

6.4 Tools for Finding Root Causes and Solutions

...any sophisticated statistical analysis should always be supplemented with easily understood graphics that can be evaluated by people who can relate the conclusions to the process; people who may lack training in advanced statistical analysis.

Thomas Pyzdek, 1992

As you collect data to investigate assignable causes and potential improvements, you will often face the need to sort through and understand the information you obtain. This involves organizing and summarizing your data and looking for patterns, trends, and relationships. Tools such as scatter diagrams, run charts, cause-and-effect diagrams, histograms, bar charts, and Pareto charts can all help you here. These tools are described briefly below and illustrated in greater detail in the pages that follow.

- Scatter diagrams display empirically observed relationships between two process characteristics. A pattern in the plotted points may suggest that the two factors are associated, perhaps with a cause-effect relationship. When the conditions warrant (i.e., a constant system of chance causes), scatter diagrams are natural precursors to regression analyses that reveal more precise information about interrelationships in the data.
- Run charts are a specialized, time-sequenced form of scatter diagram that can be used to examine data quickly and informally for trends or other patterns that occur over time. They look much like control charts, but without the control limits and center line.
- Cause-and-effect diagrams (also know as Ishikawa charts) allow you to probe for, map, and prioritize a set of factors that are thought to affect a particular process, problem, or outcome. They are especially helpful in eliciting and organizing information from people who work within a process and know what might be causing it to perform the way it does.
- Histograms are displays of empirically observed distributions. They show the frequencies of events that have occurred over a given set of observations and period of time. Histograms can be used to characterize the observed values of almost any product or process attribute. Examples include module size, defect repair time, time between failures, defects found per test or inspection, and daily backlogs. Histograms can be helpful for revealing differences that have taken place across processes, projects, or times.
- Bar charts are similar in many ways to histograms, but they need not be based on measures of continuous variables or frequency counts.
- Pareto charts are a special form of histogram or bar chart. They help focus investigations and solution finding by ranking problems, causes, or actions in terms of their amounts, frequencies of occurrence, or economic consequences.

The sections that follow give brief descriptions of these analytical tools and techniques. More complete illustrations, albeit in nonsoftware settings, can be found in several references [Ishikawa 86, Brassard 88, Brassard 89, Montgomery 96]. We particularly commend the 1986 revised edition of Ishikawa's *Guide to Quality Control* [Ishikawa 86].

The Venn diagram in Figure 6-6 shows where the tools described in the following pages fit relative to the fact-finding and analysis activities that lead to identifying root causes and potential solutions [Brassard 88, 89].

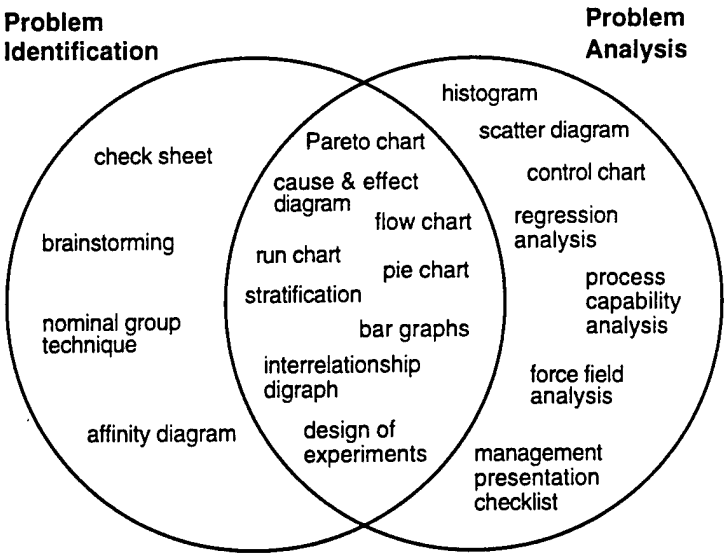


Figure 6-6: Application Areas for Analytic Tools

Scatter Diagrams

Description

A scatter diagram is a plot of observed values that shows how one variable has behaved relative to another. Scatter diagrams are often used as a first step in the exploration of data, especially as part of a search for cause-effect relationships. Sometimes there is an assumption that one variable is "dependent" and the other "independent," but this does not have to be the case.

Scatter diagrams are used to address questions such as "Does company A's product work better than company B's?", "Does the length of training have anything to do with the number of defects an engineer injects?", "Are there any obvious trends in the data?", and so forth. When a scatter diagram suggests that a relationship may exist between two variables, its use is often followed by more formal statistical methods such as exploratory data analysis or regression analysis.

Scatter diagrams (sometimes called scatter plots) are limited in that they usually deal with only two variables at a time. This constraint exists because the results are displayed on sheets of paper or CRT screens, both of which are two-dimensional media. When you have reason to investigate more than two dimensions at a time, you will likely want to use more formal (statistical) methods that facilitate multivariate analyses. Here, though, the pitfalls are many, and it is easy for amateurs to go astray. We strongly suggest that you solicit the assistance of a competent statistician. Furthermore, we urge you to do this before the data are collected. Good statisticians know things that will help you get data in ways that will help you reach valid conclusions economically.

Example

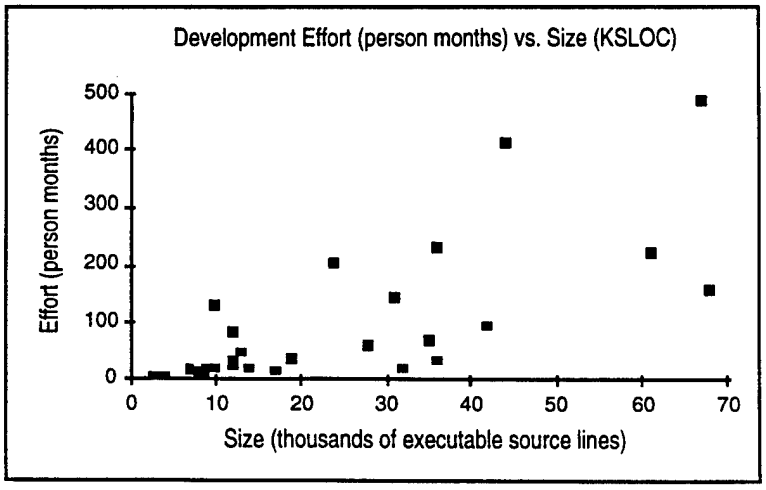


Figure 6-7: Example of a Scatter Diagram

Run Charts

Description

Run charts are plots of individual values arranged in a time sequence. They can be used to monitor a process to see if trends are apparent or if the behavior of the process is changing in other ways. Observed values of production throughput, product size, team size, number of defects found, backlogs, and cumulative or daily resource consumption are all candidates for run charts. Run charts can visually display the behavior of any interval- or ratio-scale variable.

One danger when using a run chart is the tendency to see every variation in the plotted values as being important. The control charts that have been illustrated in Chapter 5—with their control limits, run tests, and formal tests for unusual patterns—were developed specifically to counter this tendency. In one sense, run charts are nothing other than precursors to control charts. But run charts can also be used when trends are known to exist, so that control charts (in their standard form) do not apply. Although interpreting run charts can be risky without a formal methodology, it is often possible to relate sudden shifts in plotted values or trends to specific events. When these relationships exist, it is helpful to annotate the run chart to indicate the events.

Run charts, like Pareto charts and histograms, refer to events that occurred in a particular time period. Thus, they should show the time period covered, the frequency of measurement (if appropriate), and the unit of measurement used. Center lines such as medians of means (averages) are sometimes shown, but this can induce misinterpretation when the process that produced the data is not under statistical control.

Examples

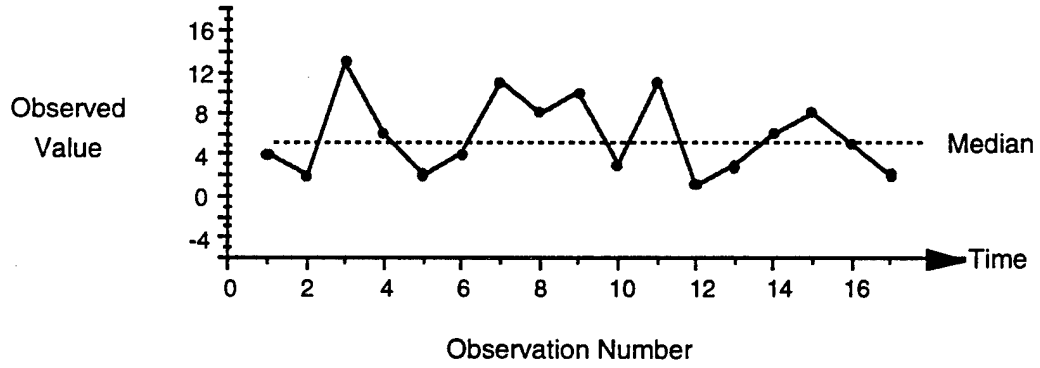


Figure 6-8: Example of a Run Chart with Level Performance

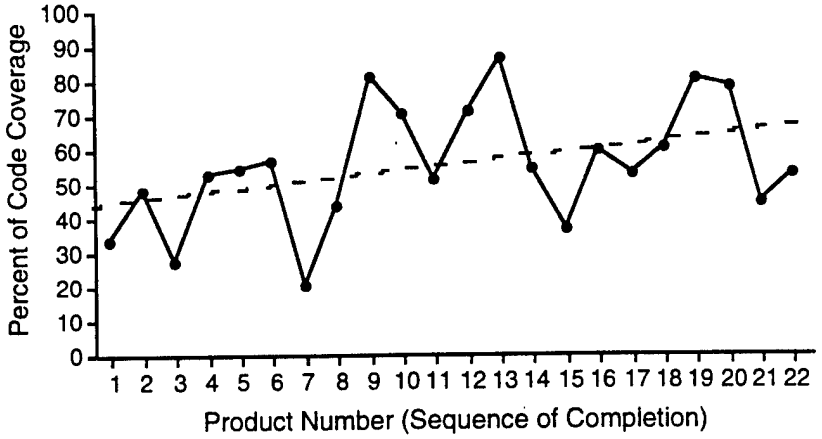


Figure 6-9: Example of a Run Chart with a Rising Trend

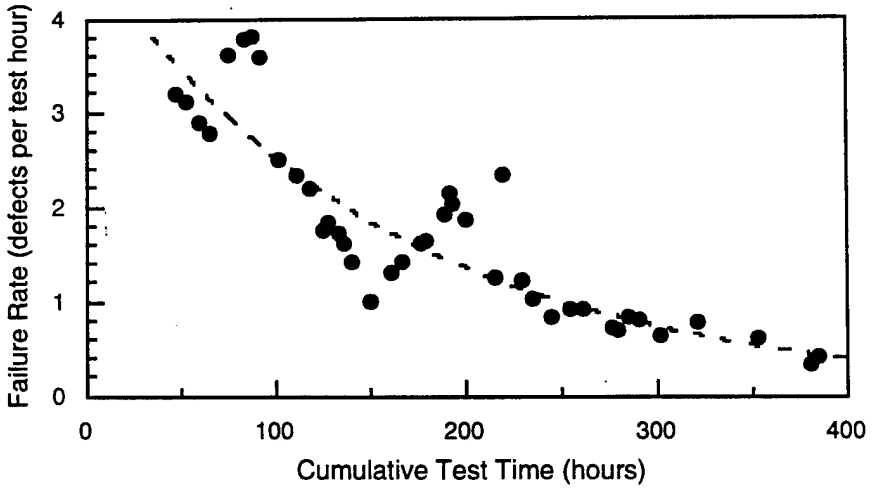


Figure 6-10: Example of a Run Chart with a Falling Trend

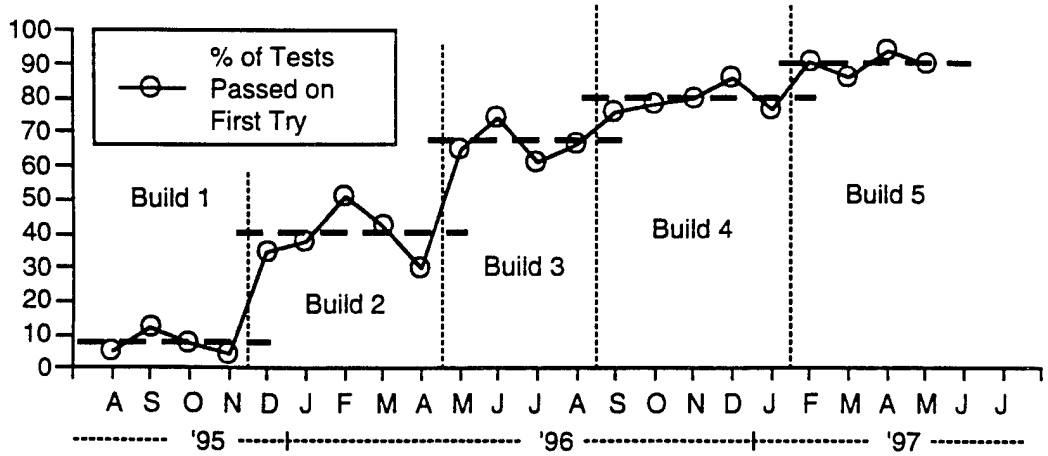


Figure 6-11: Example of a Sequential Run Chart

Cause-and-Effect Diagrams

Description

A cause-and-effect diagram is a graphical display that is used to probe for and show relationships between a problem (the effect) and its possible causes. They are often called Ishikawa charts, after the man who originated them in 1943 [Ishikawa 86]. They are also called fishbone charts, due to their visual similarity to the skeleton of a fish.

When cause-and-effect diagrams are used to explore the behavior of a process, it is best if the diagrams are assembled by people who actually work in the process. When it comes to pinpointing problems and searching for root causes, there is no substitute for first-hand knowledge.

It is also wise to have people expert in different parts of the process participate. Diagrams drawn by one or two people are apt to be ineffective because they lack a sufficiently broad base of observational experience. For this reason, cause-and-effect diagrams are often drawn during brainstorming sessions that include people with differing viewpoints.

Although cause-and-effect diagrams are (initially) subjective, they can be based on (and annotated to show) factual information such as measured values and dates of occurrences.

Cause-and-effect (CE) diagrams can be divided into three types [Ishikawa 86]:

1. the dispersion analysis type
2. the production process classification type
3. the cause enumeration type

The dispersion analysis type of cause-and-effect diagram is constructed by repeatedly asking the question, "Why does this dispersion (i.e., scatter) occur?" Its strong point is that it helps organize and relate factors that cause variability in products and other process outcomes. Its weak points are that the form of the resulting diagram is dependent on the views of the people making it, and that small causes may not get isolated or observed.

The production process classification type of CE diagram is constructed by stepping mentally through the production process. This may be done in one of two ways: by making the steps in the process the major ribs of a fishbone diagram, or by superimposing boxes on the backbone so that each box is a step in the production process. When the process steps are displayed along the backbone as in Case 2 (illustrated in Figure 6-12), the causes are depicted on lines (ribs) that feed into either a box or one of the backbone segments that connects sequential boxes. The strength of this type of diagram is that it is easy to assemble and understand. The weaknesses are that similar causes often appear in more than one place and that problems resulting from combinations of more than one factor are difficult to illustrate.

The cause enumeration type of CE diagram is generated by listing all possible causes and then organizing the causes to show their relationships to the aspect of product or process quality that is being examined [Ishikawa 86]. Figure 6-12 shows a simple example. This type of cause-and-effect diagram can also be produced in brainstorming sessions where principal categories such as manpower, materials (inputs), methods, and machinery (tools) are used to prompt probing questions that uncover possible causes. The completed diagram may end up looking much like one produced by the dispersion analysis process, but it may not. The thought processes used to generate cause enumeration charts are (and should be) more free form and less constrained than for dispersion analysis charts. The strength of the cause enumeration chart is that enumerating large numbers of likely causes reduces the probability of overlooking a major problem area. When done well, this tends to give a more complete picture than a chart produced by dispersion analysis. The weakness is that it may be hard to relate the twigs of the tree to the end result, which can make the diagram difficult to draw and to interpret.

Whatever method you use for producing cause-and-effect diagrams, be on the lookout for diagrams with many major ribs and few twigs. This almost always indicates that either the understanding of the process was shallow, or the diagram is too generalized. Use care also when a diagram lists only five or six causes. This kind of diagram is usually inadequately penetrating, even though its form may be correct.

Examples

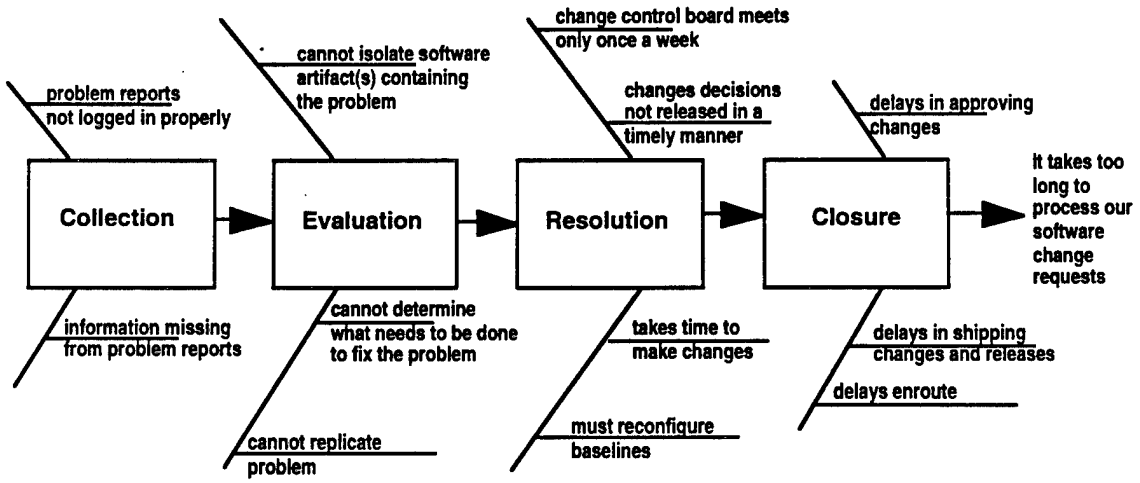


Figure 6-12: A Process Classification Cause-and-Effect Diagram

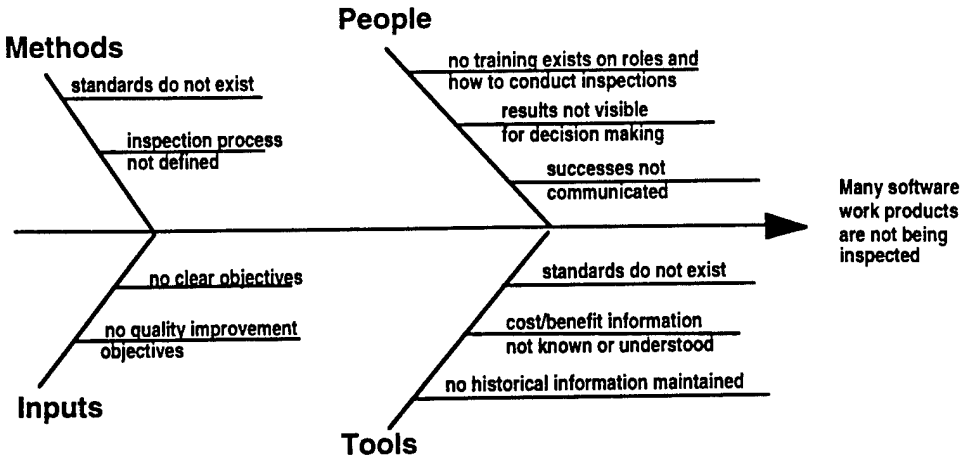


Figure 6-13: A Cause Enumeration Cause-and-Effect Diagram

Histograms

Description

Histograms take measurement data and display the distribution of the observed values. They are created by grouping the results of measurement into "cells" and then counting the number in each cell. The cells are non-overlapping, equal-width intervals along some continuous scale. The heights of the bars in histograms are proportional to the number of occurrences within each cell.

Histograms display frequency counts in ways that make it easy to compare distributions and see central tendencies and dispersions. As we have seen in Chapter 5, histograms are useful for investigating and summarizing the performance of a process with respect to the specification limits that the process or its products must satisfy.

Histograms can be helpful troubleshooting aids. Comparisons between histograms from different subprocesses, operators, vendors, or periods of time often provide insights that point to potential causes, trends, or needs for stratification. For example, twin peaks may indicate that the data have come from two different sources, each with its own distinct characteristics. If so, control charting and other interpretive or predictive applications would not be appropriate until the data have been stratified according to source.

Example

Figure 6-14: A Simple Histogram for Continuous Data

Guidelines

When constructing histograms, the cells should be placed immediately adjacent to each other (there are no gaps between cells for continuous data). Care should be used in selecting both the width and the number of cells. Ott and Schilling offer these guidelines [Ott 90]:

1. Make all cells of equal width. (The human mind tends to use the areas within cells as the basis for evaluations. Cells that are wider than others give disproportionate weight to the values represented by their height.)
2. Choose the cell boundaries so that they fall halfway between two possible observations (i.e., halfway between two adjacent values on the measurement scale that is being used).
3. Do not use too many cells. The number of cells for very large samples (e.g., 1000 or more) should be between 13 and 20. When the number of samples is not very large, Sturges' rule of thumb provides the following guideline for the relationship between the number of cells c and the sample size n:

   $$c = 1 + 3.3 \log_{10} n$$

   This leads to the table in Figure 6-15.

Sample size n	Number of cells c
6–11	4
12–23	5
24–46	6
47–93	7
94–187	8
188–376	9
377–756	10
757–1519	11
1520–3053	12
3054–6135	13
6136–12,388	14
12,329–24770	15

Figure 6-15: The Number of Cells to Use When Plotting Histograms

Since the relationship between the number of cells (c), the number of data points (n), and the cell width (Δ) is $c = \frac{\Delta}{n}$, compromise will sometimes be needed if the objectives of guidelines 2 and 3 above are to be satisfied simultaneously.

Ishikawa offers an alternative (and simpler) set of guidelines. His recommendations are shown in Figure 6-16 [Ishikawa 86].

Number of Observations	Number of Cells
under 50	5–7
50–100	6–10
100–250	7–12
over 250	10–20

Figure 6-16: Ishikawa's Recommendations for Number of Histogram Cells

Bar Charts

Description

Bar charts, like histograms, are used to investigate the shape of a data set. They are similar in many respects to histograms, but defined on sets of discrete values. Because of this, they can display any numerical value, not just counts or relative frequencies. Thus bar charts can be used to display data such as the total size, cost, or elapsed time associated with individual entities or with sets of products or process steps.

Because bar charts are defined on discrete scales, cell width is irrelevant, and there are always gaps between cells. You are free to use bars of any width you like. Bars of different widths, though, should be used only as one of several possible ways to distinguish between different data sets (coloring, shading, staggering, or labeling are usually preferable).

The concepts of average and standard deviation have no meaning for the independent variable in bar charts that are defined on discrete scales whose values are not equally spaced on the real number line. Medians, modes, and ranges, however, can be used with ordinal scales, even though the distance between cells has no meaning.

Example

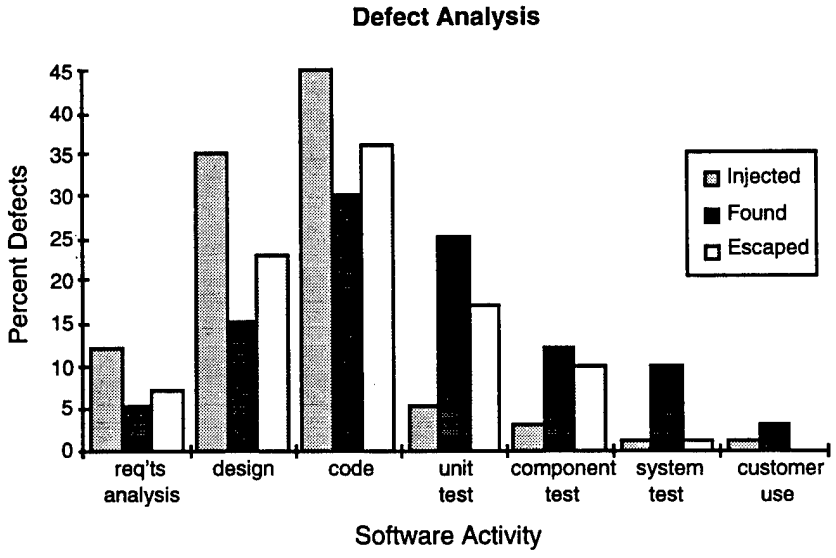


Figure 6-17: A Bar Chart That Compares Three Discrete Distributions

Pareto Charts

> *In case you are not familiar with Pareto, he was an Italian economist who developed a rule describing wealth distribution. It ended up being a fairly useless statistic on its own, but its misapplication by 80% of its users has caused more that 20% of quality failures in the world.*
>
> *Charles A. Barclay, in a posting to bit.listserv.quality, 1997*

Description

Pareto analysis is a process for ranking causes, alternatives, or outcomes to help determine which should be pursued as high priority actions or opportunities for improvement. It is a useful technique for separating the "vital few" from the "trivial many" [Juran 88]. Pareto charts can be used at various stages in a quality improvement program to help select the steps to take next. They can also be used to address questions such as, "On which types of defect should we concentrate our efforts?" and "What parts of the process are the biggest contributors to the problem we are examining?"

In their simplest forms, Pareto charts are essentially frequency counts or amounts displayed in descending order. In more sophisticated analyses, Pareto charts may rank causal factors or potential actions in decreasing order of their estimated economic costs or consequences.

Pareto charts may seem simple, and they are. Nevertheless, the benefits that this kind of visual display has over a table of numbers should not be underrated. As those who have succeeded in quality improvement can attest, much of the battle lies in getting everyone in a group—employees and managers alike—to share a common view of the problems faced and the actions needed.[25] Pareto charts help to create this common view.

As simple as Pareto charts are, there are some points that should not be overlooked. In particular, the frequencies or other values plotted on a Pareto chart almost always have a time period associated with them. This time period should be made explicit, so that viewers of the chart can attach correct interpretations to the rankings that are displayed.

One effective use of Pareto charts is to compare the before and after states related to improvement actions. This can give a quick visual appreciation of the effectiveness and progress (or lack thereof) related to actions that have been taken. Formal methods exist for testing whether or not differences between successive Pareto charts may be due solely to chance, so that inappropriate conclusions will not be drawn [Kenett 91].

---

[25] Everyone "knows" that it is easier and more consequential to reduce a tall bar by half than to reduce a short bar to zero.

One caution: if the processes that produced the data are not stable, Pareto charts easily lead to erroneous conclusions and improper actions. This is especially true when the chart shows frequencies of occurrence for different types of problems, some of which may have assignable causes. Assignable causes mean uncontrolled variation, and these causes come and go. Frequencies associated with assignable causes have little meaning, since there is no single distribution that underlies them. When a process is not in statistical control, the Pareto chart can easily look radically different from month to month, even when no action is taken.

Example

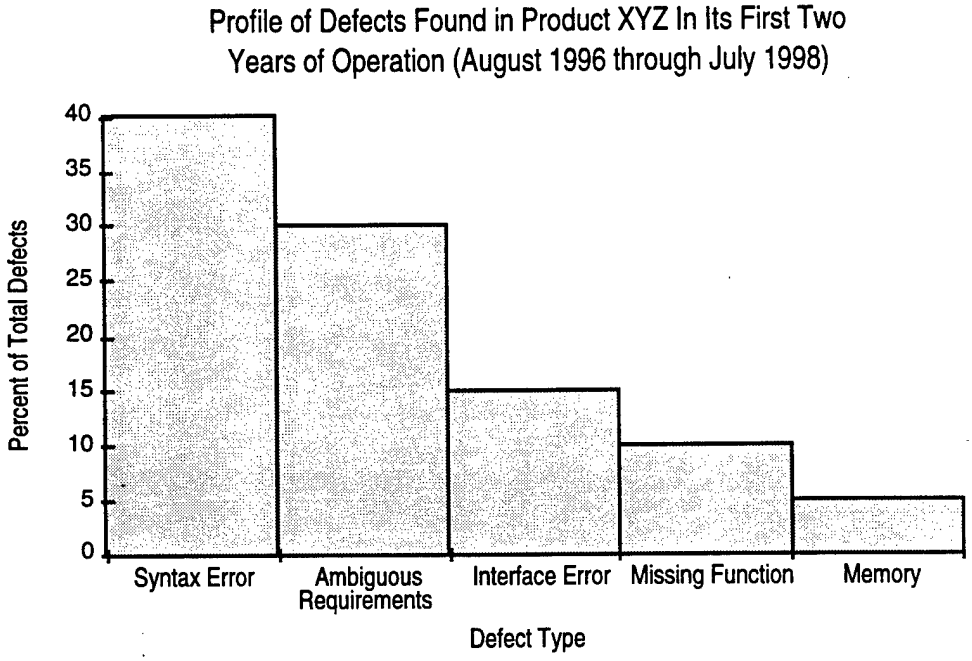


Figure 6-18: Example of a Pareto Chart

6.5 Technologies and Methodologies for Changing or Improving Software Processes

Statistical quality control is 90% engineering and only 10% statistics.[26]

H. F. Dodge, as quoted by Grant and Leavenworth, 1996

The methods and technologies in the list below are examples of things that various software organizations have found useful for improving the processes they use to develop and support software systems. Many of the items in the list are discussed in more detail in a survey published by Austin and Paulish [Austin 94]. Each chapter of that report is augmented with extensive reference lists to guide further reading on these topics.

Cleanroom software development. Cleanroom is a software production method that originated in the Federal Systems Division of IBM in the late 1970s and early 1980s. It combines practices of formal specification, nonexecution-based program development, incremental development, and independent statistical testing.

Computer aided software engineering (CASE). CASE is a collection of methods that use software tools for automating parts of software development and support processes.

Defect prevention process (DPP). This method, sometimes referred to as causal analysis, was pioneered by IBM in the 1980s. It assists in categorizing defects so that they can be removed systematically and avoided in future software development projects.

Estimation. Estimation is a system of measurements, tools, practices, and skills for predicting characteristics of software projects before the project begins. Criteria for establishing, operating, and sustaining good estimating processes have been published in checklist formats by Park [Park 95, 96b].

Formal inspection. Formal inspections were pioneered by Michael Fagan at IBM in the 1970s. They provide a structured technique for examining software artifacts for defects prior to their use.

Interdisciplinary group methods (IGMs). This is a collection of methods that encompass various forms of planned interaction engaged in by people of

[26]Harold Dodge's statement was not intended to belittle the importance of statistics. The point he was making is that once statistics such as control charts point to the need to hunt for trouble, 90% of the hard work is yet to be done. The real payoffs from statistical quality control lie in the hunt for—and elimination of—assignable causes of nonrandom variation and misaligned processes.

diverse expertise and functional responsibilities working together as a team toward the completion of a software system. Example methods include nominal group technique (NGT), joint application design (JAD), groupware, group decision support systems (GDSS), quality circles, and concurrent engineering.

ISO 9000 certification. ISO 9000 is a series of standards established by the International Standards Organization (ISO) for certifying that an organization's practices provide an acceptable level of conformance to methods that have been found to produce quality products.

Personal Software Process℠ (PSP℠).[27] PSP is a structured set of process descriptions, measurements, and methods that can help engineers improve their individual performance. It provides forms, guidelines, and procedures to assist engineers in estimating and planning their work. One of the basic principles of the PSP is, "If you don't put a quality product into testing, you won't get a quality product out of test." PSP training focuses on showing engineers why this is true and how they can take steps to ensure that their product is of high quality before they start to test it [Humphrey 95].

Process definition. Process definition uses structured methods to formally specify or model software development processes in ways that foster communication, training, repeatable operation, and analysis.

Quality function deployment (QFD). QFD is a family of tabular and graphical methods that can be used to help define software functional requirements. It emphasizes meeting customer needs, distinguishing results from those of competitors, and accounting for implementation difficulty.

Software Measurement. Software measurement is a collection of methods that provides timely quantitative information for defining, tracking, and improving project and process performance.

Software process assessment (SPA). This is a formal method for assessing software development organizations to determine the strengths and weaknesses. Results typically include structured ratings of process maturity and prioritized recommendations for potential areas of improvement.

Software reliability engineering (SRE). SRE is a collection of methods, measurements, and models for statistically estimating and predicting failure rates of a software system. It is often used for determining when a product is ready for release.

Total quality management (TQM). This collection of methods and team-based activities is oriented towards continuously improving the quality culture of an organization and the quality and cost of its products.

---

[27] Personal Software Process and PSP are service marks of Carnegie Mellon University.

Discussions of additional technologies and methodologies for improving software process can be found in the following publications:

- *The Capability Maturity Model for Software* [Paulk 93a, 93b, 95]
- *Software Process Evolution at the SEL* [Basili 94]
- *Orthogonal Defect Classification* [Chillarege 92, 96]
- *People, Organizations and Process Improvement* [Perry 94]

7 More About Analysis and Use of Process Measures

This chapter describes additional principles and practices that are important to the successful use of measurements for managing and improving software processes.

7.1 Statistical Inference as a Basis for Action

> *You have to know when to argue with data...Data are about the past.*
>
> *Andy Grove, Intel, 1996*

When using measurements to manage and improve software processes, you will almost always be trying to predict or affect future performance. When you have these goals in mind, you will invariably find yourself in the realm of analytic studies. Understanding the implications of this will help you distinguish between valid and invalid use of measurements. It will also help you understand where responsibilities lie for making predictions come true.

The next two subsections describe the principal differences between enumerative and analytic studies and explain what the differences mean in practice. Textbooks in statistics have been slow to give the distinctions between these studies the attention they deserve [Hahn 93]. Perhaps because of this, the advice you find here may seem unfamiliar and excessively conservative. If you are uncomfortable with the recommendations that follow, we strongly encourage you to supplement these discussions by reading what Deming, Hahn, Meeker, and Wheeler have to say on the topic [Deming 75, Hahn 93, Wheeler 95].

Enumerative Studies

An *enumerative* study is one in which action will be taken on the materials within the frame studied. The frame defines the population that is sampled. Software engineering examples of enumerative studies include:

- inspections of code modules to detect and count existing defects
- functional or system testing of a software product to ascertain the extent to which a product has certain qualities
- measurement of software size to determine project status or the amount of software under configuration control
- measurement of staff hours expended, so that the results can be used to bill customers or track expenditures against budgets

In each case, the aim of an enumerative study is descriptive: to determine "how many" as opposed to "why so many." An enumerative study answers questions like

- How many defects were found by inspecting the product code?
- How many problem reports have been received from customers?
- How many user manuals do we have on hand?
- How many employees are there in our organization?
- What percent have been trained in object-oriented design methods?
- How large were the last five products we delivered?
- What was the average size of our code inspection teams last year?
- How many staff hours were spent on software rework last month?

Enumerative studies are not designed to guide process change or predict results that might be obtained from different sampling frames, either now or in the future.

In an enumerative study, it is possible (although perhaps not practical) to reduce errors of sampling to arbitrarily low levels. In the limit, in theory, you could measure every element in the frame and thus have no sampling error whatsoever.[28] In an analytic study, on the other hand, it is impossible to compute the risk of making a wrong decision, no matter how many data points are obtained [Deming 75].

The steps for collecting and analyzing measurement data for an enumerative study are as follows [Deming 75, Hahn 93, Park 96a]:

1. Define the goal of the study.
2. Define the characteristics or properties of interest.
3. Explicitly and precisely define the target population about which inferences are desired.
4. Define the frame from which the sample will be taken:
   - Obtain or develop a specific listing or other enumeration of the population from which samples will be selected. This population is often not identical to the target population.
   - Describe the environment in which the attributes will be measured.
5. Evaluate the differences between the sampling frame and the target population, and identify the possible effects that the differences could have on the conclusions of the study.

[28]Errors of measurement, however, can still be present, even when every element in the sampling frame is included in the sample. Rounding errors, approximations, biases, and errors in reading instruments or recording values are just a few examples.

6. Define, clearly and precisely, the attribute(s) that will be measured, the measures and scales that will be used, and the manner in which measurements will be made.
7. Design and execute a sampling procedure that supplies entities to be measured which are appropriately randomized and as representative of the target population as possible.
8. Review and assess the collected data.
9. Identify the other assumptions that will be used in drawing inferences. This includes any assumption of a specific underlying distribution (such as the normal, binomial, Poisson, or exponential distribution) and all assumptions related to the representativeness of the sampled values.
10. Ensure that the tools and methods used for analysis are consistent with the nature of the data.
11. Ensure that the users of any inferences drawn from the study are made fully aware of the limitations implicit in the assumptions that underlie the inferences.

We refer you to the standard texts on statistics for guidance on using methods such as design of experiments, regression analysis, confidence intervals, correlation, multivariate analysis, and sampling when collecting and analyzing data for an enumerative study and presenting the results.

Analytic Studies

> *Statistical theory as taught in the books is valid and leads to operationally verifiable tests and criteria for enumerative studies. Not so with an analytic problem, as the conditions of the experiment will not be duplicated in the next trial. Unfortunately, most problems in industry are analytic.*
>
> *W. Edwards Deming, 1980*

> *No amount of sophistication is going to allay the fact that all your knowledge is about the past and all your decisions are about the future.*
>
> *Ian E. Wilson*

An *analytic* study differs from an enumerative study in that action will be taken on the process (or cause system) that produced the data, not on the materials within the frame from which the data came. The aim of an analytic study is to predict or improve the behavior of the process in the future.

When we conduct an analytic study, we use data from an existing process to predict characteristics of future output or performance from either the same process or similar, perhaps modified processes. Thus, an analytic study is always concerned with a process, not with a population.

As Deming points out...

> *There is a simple criterion by which to distinguish between enumerative and analytic studies. A 100 percent sample of the frame provides the complete answer to the question posed for an enumerative problem, subject of course to the limitation of the method of investigation. In contrast, a 100 percent sample of a group of patients, or a section of land, or of last week's product, industrial or agricultural, is still inconclusive in an analytic problem.*
>
> *W. Edwards Deming, 1975*

Most analyses of measurements to support process management are analytic studies, and much of this guidebook focuses on the analytic aspect of statistical inference. This is not to say that enumerative studies are never used. They are often used to provide status reports that point to the need for management action. They can also be used to establish relationships among existing processes or products. Examples of analytical studies, on the other hand, include

- evaluating software tools, technologies, or methods—for the purpose of selecting among them for future use
- tracking defect discovery rates to predict product release dates
- evaluating defect discovery profiles to identify focal areas for process improvement
- predicting schedules, costs, or operational reliability
- using control charts to stabilize and improve software processes or to assess process capability

In analytic studies, investigators are concerned with making inferences or predictions that go *beyond* the sampled data. This is inherently more complex than drawing conclusions from enumerative studies. The reason is that analytic studies require the added, often unverifiable assumption that the process about which one wishes to make inferences is statistically identical to the one from which the sample was selected [Hahn 93]. Whenever this assumption does not hold—and it almost never does—it will be impossible to make unconditional, quantitative statements about the probabilities or risks associated with future outcomes. The best that we can do is to make weak, conditional statements, subject to the assurances of subject-matter experts that *all* significant conditions in the future will be

exactly as they were when the samples were obtained. Even these statements are at risk, because all too often the samples that they are based on are convenience samples (and hence give biased views of the sampling frame), rather than representative samples of randomly occurring outcomes. Wheeler puts it this way:

> *In an Analytic Study there is no way to define a random sample of the future. All data, and all analyses of data, are historical. However, in an Analytic Study the inference of interest involves prediction, an extrapolation into the future. Probability theory offers no help here. All samples become "judgment samples," and there is no way to attach a probability to a specific future outcome.*
>
> *Donald J. Wheeler, 1995*

Extrapolations to the future *always* go beyond what statistical methods alone can deliver. In addition to statistical methods, extrapolations require a model of the effects that changes will have on processes that are studied, together with assurances of the continued existence of all conditions not covered by the model (earthquakes, fires, strikes, unforeseen problems, changes in suppliers, raw materials, personnel, environments, etc.). Only subject-matter experts or people who manage and operate these processes can provide the models and make these assurances.

So whenever you see a confidence, statistical tolerance, or prediction interval that is reported in the setting of an analytic study, you should understand that it is at best a lower (optimistic) bound on the true uncertainty [Hahn 93]. The interval that expresses the total uncertainty is almost assuredly wider, and perhaps much wider, than the statistical intervals alone might lead you to believe. Analysts owe it to the users of their studies to make this point clear. Unfortunately, in the software world as in other business endeavors, this is seldom done.

The most critical issues in any statistical study are the models used to relate the target population to the sampled population and the assumptions associated with the representativeness of the data. These models and assumptions are often implicit, and hence hidden from both analysts and users of the inferences. Departures from the (implicit) models and assumptions are common in practice. This can easily invalidate a formal statistical analysis. Failure to state and validate the models and assumptions can produce a false sense of security. This, in many instances, is the weakest link in the inference chain [Hahn 93].

Consequently, when performing analytic studies, we are motivated to use statistical methods that require as few assumptions as possible. This is one reason for the well-deserved popularity of Shewhart's control charts and 3-sigma limits. Another is that control charts also address the problem of consistency of conditions over time. When you can demonstrate that a process has operated consistently in the past, you are on much sounder ground when you

predict that the observed performance will continue into the future. The converse is also true. When a process you are examining is not in (or near) statistical control with respect to all characteristics of relevance, the applicability of statistical intervals or any other method for characterizing the process can be undermined easily by unpredicted trends, shifts, cycles, or other variations. Shewhart made this point very clear:

> *Sampling theory applies to samples arising under controlled conditions. Too much emphasis cannot be laid upon this fact. To be able to make accurate predictions from samples, we must secure control first just as to make accurate physical measurements, we must eliminate constant errors.*
>
> *Walter A. Shewhart, 1931*

The steps for collecting and analyzing measurement data for an analytic study are essentially the same as for enumerative studies. There are, however, some additional aspects that should be addressed. For instance, because we wish to draw conclusions about a process that may not even exist at the time of the study, the process that we sample is likely to differ in several ways from the process whose performance we seek to predict or describe. Moreover, instead of dealing with a set of identifiable units as in an enumerative study, we must deal with observations taken in some "representative" manner from an existing process.

What this means is that, in an analytic study, we have both greater opportunity and greater responsibility for defining the specific process to be sampled and the way the sampling will proceed [Hahn 93]. In conducting analytic studies, you should always aim to consider as broad an environment as possible. This means addressing, insofar as possible, the full range over which inputs, resources, personnel, and operating conditions might be encountered in the future. This is contrary to the traditional advice applied to most scientific investigations, where one tries to hold all variables constant except those key to the study itself.

The reason for making an analytic study broad is to narrow the gap between the sampled process and the process of interest. This is especially important if there are possible interactions between the factors under investigation and any background or contextual conditions.

Another difference between enumerative and analytic studies is that in an enumerative study, the probability model that you use to draw inferences is attached to the *selection* of the items in the sample. With such a probability model, you can work out formulas for induction and for estimating the uncertainties introduced by using a sample in place of a full census. In contrast, in an analytic study, the probability model must apply to the *behavior* of the phenomenon itself [Wheeler 95]. Since you have less control of the behavioral process in analytic studies than you do of the sampling process in enumerative studies, the assumptions on which your inferences will be based are apt to be less justifiable, and there will be more (unknown) uncertainties in the inferences you draw.

Analytic studies can also differ from enumerative studies in that the sampling may take place over time. When this is the case, it is usually advisable to sample over relatively long periods. Observations taken over short periods are less likely to be representative of either average performance or long-run variability, unless the process is in strict statistical control.

The concept of statistical control is frequently important to analytic studies. When a process is in statistical control *and remains so*, the current data can be used to draw inferences about the future performance of the process. Data from a process in strict statistical control correspond to the commonly used (and at times misused) assumption of independent and identically distributed random variables.

Many analytic studies are dynamic in nature. Here, observed values are collected over a period of time, and the results are used to guide changes to an ongoing process. The changes may be for the purpose of stabilizing the process (statistical process control), hitting a desired target (industrial process control), or improving the process (reducing variability or resetting the process to a more profitable target). In these cases, an analytic study may also include steps like the following:

1. Analyze process performance data for stability:
    a. Find out what the process is doing now.
    b. Is the process stable?
    c. If the process is not stable
        - Find all assignable causes.
        - Fix the assignable causes so that they cannot occur again.
    d. If the process is stable, do one of the following:
        - Leave it alone and, if desired, proceed to capability evaluation.
        - Introduce adjustments (or improvements) and track the results.
2. Once stability is established, determine process capability by comparing the performance of the process to the requirements it is supposed to meet.
3. Use the results of stability and capability analyses to guide and evaluate investigations and decisions.
    a. Identify improvement opportunities.
    b. Assess probabilities of future outcomes.
    c. Determine the effectiveness of improvement actions.

In analytic studies, you should be mindful that statistical inferences are inherently limited to the conceptual population of entities that could, at least in theory, have been produced and measured at the same time and in the same manner as those in the study. However, as Hahn and Meeker point out [Hahn 93]

> *...in the overwhelming majority of cases, the investigator's prime concern is not with the sampled process but with a different one. Any extrapolation of the inferences from the sampled process to some other process is generally beyond [the analyst's] area of competence. The validity of such an*

extrapolation needs to be assessed by the subject-matter expert. Under these circumstances, determining whether or not to use statistical inference methods requires a judgment call. If such methods are to be used, it is, moreover, the analyst's responsibility to clearly point out their limitations.

Gerald Hahn and William Meeker, 1993

In software settings, most studies will be aimed at predicting or guiding events that will take place in the future. Hence they will be analytic, not enumerative. Before embarking on any data collection or analysis endeavor, you should have a clear understanding of the differences between the two types of studies, so that you can design collection activities properly and not ask more of data than the data can deliver. Understanding the differences between enumerative and analytic studies will also help you explain why some questions that software managers ask are impossible to answer with statistical methods alone, and why the managers themselves must assume much of the responsibility for ensuring the effectiveness of the results.

7.2 Reviewing and Assessing Collected Data

Before you begin analyzing measurement data, there are certain criteria that the reported values must satisfy if your analyses are to have any merit or credibility. These criteria are discussed briefly below. It is important to determine whether or not the reported values satisfy the criteria, and to do this very early in the measurement process. You will also avoid unnecessary rework and improve the reliability of analyses if you keep these criteria in mind when selecting and defining data elements to be measured and the processes you will use for collecting, recording, and retaining measurement results.

Criterion 1: Verified

Verified data are data that have been examined to assure that they have been collected according to specifications and contain no errors. Typically, the examination ascertains that the reported values are

- **of the correct type** (i.e., numeric, alphanumeric). Often, some data elements can be predetermined to be numeric only. Other data elements may be limited to the use of certain characters or symbols. The examination for verity ascertains that collected and reported data are consistent with such specifications.
- **in the correct format.** Nearly all data elements will have specified formats. Dates, monetary values, counts, product names, process names, product IDs, job codes, tools, and priorities are typical examples of data that will have specified formats. In these cases, the values that are reported must be verified to be in the expected formats.

- **within specified ranges.** Many kinds of collected data can be examined for valid ranges of values. Valid ranges can be lists of acceptable names or numerical ranges of acceptable dates or values. Impossible values should be investigated and corrected before they contaminate a database.
- **complete.** Measurement data must contain the essential data elements and the associated definitions and contextual information that are needed to understand and interpret the data values. For example, each reported value should be identified in terms of the entity measured, time of occurrence, time of collection, collector, definition, and measurement tools used.
- **arithmetically correct.** If the collected data contain values that result from arithmetic operations, the examination should verify that the arithmetic operations have been performed correctly.
- **internally and externally consistent.** Consistency is difficult to determine, since it implies that the examiner is sufficiently knowledgeable of previously reported data to be able to make this determination. Nevertheless, it is important that outlandish or improbable data elements be investigated and their correctness verified if erroneous analyses are to be avoided. This may involve contacting the source of the data to confirm the correctness of the reported values. It may also involve understanding and recording the circumstances associated with the occurrence of the value, so that others can judge later whether or not the data should be used in an analysis.

Criterion 2: Synchronous

You can think of measurements as being synchronous when the values for two or more attributes are related with respect to the time of their occurrence. The notion of synchronized measurements is particularly important when measuring attributes of a process or when using attributes of products and resources to describe the performance of a process.

Measures that are based on arbitrary time frames are particularly susceptible to problems with synchronicity. For example, productivity rates that are computed by comparing outputs to inputs over a period of time can easily be misleading if the resources actually expended are not appropriately matched to the products produced or the period measured. If the time to execute the process is not considered, lags within a process may mean that output statistics do not correspond to input statistics (i.e., there is not a valid cause-effect relationship). The ratios of outputs to inputs that get used for productivity measures may then have little significance.

One frequently encountered situation where unsynchronized data occur is when effort reports are prepared on a monthly basis while size and progress measures are reported at weekly intervals.

Criterion 3: Self Consistent

Inconsistencies in the way measurements get defined or collected can lead to flawed analyses. Examples include

- values associated with accounting months that become intermixed or confused with values that apply to calendar months
- effort expenditures reported in terms of calendar months that contain differing numbers of working days
- work breakdown structures, measurement definitions, and process definitions that change from project to project
- changes to personnel or job descriptions that cause reclassifications of effort categories or tasks performed

No value should be entered into a database until it is confirmed that its definition is consistent with other recorded values of the same type. Self consistency is essentially an apples-to-apples issue. If the ground rules for measurements shift with time or from project to project, it will be very difficult to compare one measured value to another. This in turn will make it difficult to attach significance to observed changes or trends in measured results.

When data you are recording are inconsistent with data that exist, you should ensure that the two types are kept distinct and that the definitions for each are recorded.

Criterion 4: Valid

> *It is a common human weakness to imagine that because a metric is intended to measure something, it actually does!*
>
> *— source unknown*

At the most basic level, you must be able to demonstrate that the values used to describe an attribute truly describe the attribute of interest. For this to be true, the measurements must be well defined. In particular, the rules for measuring must be stated explicitly, and this must be done in ways such that there are no questions as to what the rules are. Any decisions left to local interpretation or to the judgment of data collectors must at worst lead to only immaterial differences in measured values. These points are discussed further in Section 8.3.

Requiring that measured values match their definitions does not mean that every project must use exactly the same definitions. But it does mean that the definitions must be stated and recorded explicitly and then communicated explicitly to all who collect or use the measurement results. Without adequate and consistent definitions at all points in the measurement process, no data can be considered to validly represent what they purport to represent.

7.3 Assessing Predictive Validity

> *When a process is out of control...we cannot begin to define a meaningful probability distribution function, f(x), and the notion of the mean and variance of the distribution of X vanishes. Thus, while we may always calculate statistics from the data generated by an out-of-control process, and while these statistics will still describe the past data after a fashion, they will not be predictive. Such statistics cannot be used to estimate parameters of a distribution function because the notion of a distribution function for this process is no longer well-defined.*
>
> *Donald J. Wheeler, 1995*

> *No universe exists until control is established.*
>
> *Walter A. Shewhart, 1939*

We are also concerned with validity in another sense: the validity of a prediction system, estimating process, or model. Here validity can be judged only in the context of the purpose for which the measurement results are used. Although validity of this kind may be suggested by theory, it can be tested only by empirical means, such as through comparisons of predicted results with subsequently observed values. When differences occur, you must determine the extent to which the model that is being used differs from the observed process and the extent to which the observed process differs from the expected (future) process. Shewhart's control charts can be of great help here, since they help guide judgments about the reasonableness of extrapolating observed performance to future scenarios. In particular, if a process is not stable, there is little basis for assuming that tomorrow's results will be like those of the observed process. In fact, many people would say that you have not one process then, but several. In that case, no extrapolation is likely to be statistically supportable.

Two things are worth noting here. First, predictive validity cannot be determined solely by looking at the data. Claims of predictability always require empirical verification (i.e., does the predicting method have a record of success?) [Wheeler 95]. Second, predictive validity can be affected strongly by the specific definitions used for a measure. This is especially true when decisions of inclusion and exclusion are to be made. For example, how one counts code, what code is counted, what staff hours are counted, and what problems or defects are counted can have marked effects on the abilities of models that use these data as a basis for predicting future outcomes. Checking for predictive validity must include careful checking of both the static and operational portions of measurement definitions to ensure that the rules followed are fit for the purposes intended. This must then be followed up by testing the predictive system to see whether or not the results it predicts are born out by subsequent experience.

7.4 Control Limits

> *Through the use of the scientific method, extended to take account of modern statistical concepts, it has been found possible to set up limits within which the results of routine efforts must lie if they are to be economical.*
>
> *Walter A. Shewhart, 1931*

This section outlines the rationale and role of control limits, as developed and explained by Shewhart [Shewhart 31]. It discusses the relationship of control limits to the central limit theorem and the normal distribution.

Why 3 Sigma?

> *The calculations that show where to place the control limits on a chart have their basis in the theory of probability. It would nevertheless be wrong to attach any particular figure to the probability that a statistical signal for detection of a special cause could be wrong, or that the chart could fail to send a signal when a special cause exists. The reason is that no process, except in artificial demonstrations by use of random numbers, is steady, unwavering.*
>
> *W. Edwards Deming, 1986*

In his classic texts *Economic Control of Quality* and *Statistical Method from the Viewpoint of Quality Control*, Shewhart identifies the need to establish efficient methods for detecting the presence of variability that is not due simply to chance [Shewhart 31, 39]. He also points out that the choice of limits must be taken from empirical evidence—that is, it must be based on what works well in practice.

Shewhart uses the expression $\overline{X} \pm t\sigma_X$ to characterize tolerance limits for variability due to chance causes.[29] He argues that, in practice, the method for establishing tolerance limits for observed data depends on statistical theory to furnish estimates for the mean $\overline{X}$ and standard deviation σ_X of the underlying distribution, but requires empirical experience to justify the choice of t. Shewhart states that experience indicates that $t = 3$ provides acceptable economic value. By *economic* he means that the costs resulting from missing assigned causes are balanced against the costs of looking for assigned causes when none

[29] In Shewhart's notation, boldface type designates parameters of underlying distributions. Symbols for statistics calculated from measurements are printed in standard typefaces. Although it is not as apparent as we would like, the symbol σ_x is printed in boldface type throughout this section.

exist (false alarms). In his experience, 3-sigma tolerance limits, when used for control, provide an effective criterion for discriminating between common cause variability and assignable cause variability, regardless of the underlying distribution.

Shewhart was guided to his view in part by Tchebycheff's inequality. This inequality says that when a process is stable with respect to a measured attribute X, the probability $P_{t\sigma_X}$ that any particular future observation will lie within the limits $\overline{X} \pm t\sigma_X$ is greater than

$$1 - \frac{1}{t^2}.$$

For $t = 3$, this says that at least 88.9% of all observations will fall, in the long run, within three standard deviations of the mean of the underlying distribution. The probability of false alarms will then be less than 0.111. Of course, this still leaves open the problem of obtaining good estimates for the mean and the standard deviation—topics that Shewhart and others have explored extensively.

Tchebycheff's inequality is important because it applies for any distribution. With it, there is no need to make additional assumptions—such as assuming normally distributed variables—which are so difficult to validate in practice.

When we do know something about the underlying distribution, we can be more precise in our statements about the probabilities associated with 3-sigma limits. For example, when a distribution has only one mode, when the mode is the same as the arithmetic mean (i.e., the expected value), and when the frequencies decline continuously on both sides of the mode, then the Camp-Meidell inequality applies. The Camp-Meidell inequality is similar to Tchebycheff's, but more exact. It says that the probability that a randomly selected future value will lie within the limits $\overline{X} \pm t\sigma_X$ is greater than

$$1 - \frac{1}{2.25t^2}.$$

This means that when the Camp-Meidell conditions hold, and as long as the underlying distribution remains unchanged, the probability of false alarms with 3-sigma limits will be less than 0.049.

If you are willing to assume that your measured values follow a normal distribution, you can be even more precise. With normality, the probability $P_{3\sigma_X}$ that a given value will fall within $\overline{X} \pm 3\sigma_X$ is 0.9973.[30] Therefore, when an attribute that is measured has an observed value

---

[30]Be careful of your interpretations here. Just as with the Tchebycheff and Camp-Meidell inequalities, the probabilistic statement applies to randomly selected single observations only. The probability that one or more points in a sequence of observations will fall outside the limits is considerably higher. For instance, even in a normally distributed process, the chances are greater than 50-50 that at least one point in the next 260 will fall outside the 3-sigma limits.

of X that is outside the 3-sigma limits, the probability is only 0.0027 that—for this single instance alone—this is *not* the result of an assignable cause.[31]

Note though, that in real-world settings, we can never prove that a distribution is normal. Hence the exact probabilities associated with any limits will always be at least somewhat in doubt. This, coupled with the robustness of Tchebycheff's inequality and with empirically related economic considerations, is the reason Shewhart and many others came to eschew probability-based limits and to prefer 3-sigma limits instead.

To test the consequences of this preference, Shewhart, Wheeler, and others have investigated the effectiveness of 3-sigma limits when the underlying distribution is not normal [Shewhart 31, Wheeler 92, Burr 67]. Wheeler, for example, graphically illustrates the results of simulation studies in which 1000 observations were drawn from each of 6 different distributions: uniform, right triangular, normal, Burr, chi-square, and exponential. His objective was to determine the extent to which 1-, 2-, and 3-sigma limits spanned the sets of data drawn from the six distributions. The sampling plan for each distribution was based on a random number generator, so the processes that produced the data were (*a priori*) in statistical control.

The studies showed that using 3-sigma values for control limits has little practical effect on the reliability with which the limits serve to identify uncontrolled variation for the distributions that were examined. Wheeler generalizes these results and combines them with other experience in the three-part empirical rule shown in Figure 7-1 [Wheeler 92]:

The Empirical Rule: Given a homogeneous set of data[32]

Part 1:	Roughly 60% to 70% of the data will be located within a distance of 1 sigma unit on either side of the average.
Part 2:	Roughly 90% to 98% of the data will be located within a distance of 2 sigma units on either side of the average.
Part 3:	Roughly 99% to 100% of the data will be located within a distance of 3 sigma units on either side of the average.

Figure 7-1: An Empirical Rule for the Dispersion of Data Produced by a Constant System of Chance Causes

[31]These computations all assume that you know exactly what $\overline{X}$ and σ_x are. If you must estimate $\overline{X}$ and σ_x from the data, the probabilities will be different.

[32]Homogeneous means that the system of chance causes is constant. For data generated by a process, this means that the process must be in statistical control. Data from processes that are not in statistical control cannot be characterized by any probability distribution.

Part 3 of Wheeler's empirical rule shows why 3-sigma limits result in very few false alarms, regardless of the distribution. It also explains why points that fall outside the limits are highly likely to have assignable causes. When we couple these empirical observations with the information provided by Tchebycheff's inequality and the Camp-Meidell inequality, it is safe to say that 3-sigma limits will never cause an excessive rate of false alarms, even when the underlying distribution is distinctly nonnormal.

Thus, any value for a measured attribute that falls outside 3-sigma limits sends a strong signal that investigation is warranted. Regardless of the underlying distribution, the event is sufficiently unusual to suggest that it has an assignable cause and that the process is not stable. This means that 3-sigma limits provide a robust criterion that works effectively for any underlying distribution. You do not need to make assumptions of normality to use control charts to your advantage.[33]

The Central Limit Theorem and the Role of the Normal Distribution

> *It is true that some books on the statistical control of quality and many training manuals for teaching control charts show a graph of the normal curve and proportions of areas thereunder. Such tables and charts are misleading and derail effective study and use of control charts.*
>
> *W. Edwards Deming, 1986*

We raise the issue of normal distributions and the central limit theorem because of an apparent misunderstanding by many people that data must be normally distributed to use control charts. This is not the case.

Assumptions of normality come into play when the factors used to adjust statistics for bias and compute control chart limits are calculated. For example, the constants A_2, D_3, and D_4 that are used to compute control limits for X-bar and R charts are based on the assumption that the chance causes come from a normal probability distribution.[34] Irving Burr, though, has examined 26 distributions and shown that this use of the normal probability model does not lead to values that are appreciably different from the constants that should be used for other distributions [Burr 67]. Hence, using the usual, tabulated constants with nonnormal distributions has little affect on the location of the limits or their effectiveness for detecting unusual events.

---

[33] If you reduce your limits to 2-sigma values to make control charts more sensitive, as some people do, you lose much of this protection.

[34] The tables in the appendices of Wheeler's books show how these factors have been computed [Wheeler 92, 95].

Some people mistakenly believe that control charts work only when the Central Limit Theorem applies, and many writers have propagated this misunderstanding.[35] It is true that the theorem is relevant to many situations where data are grouped and averaged. In these cases, it is often valid to treat averages of observed values as at least approximately normally distributed, regardless of the underlying distribution. However, it is not necessary to make this assumption. As we illustrated previously, the normal distribution has no real bearing on the effectiveness of 3-sigma limits, because nearly all values will fall within these limits for any process that is in statistical control, whatever the distribution. The conservative nature of 3-sigma limits makes the Central Limit Theorem irrelevant. It is especially important to understand this when dealing with data obtained "one at a time" (i.e., subgroups of size one). Sometimes there are mistaken tendencies to forcibly average these data somehow, just to appeal to the Central Limit Theorem. Better and more straightforward procedures, such as XmR charts, are almost always available.

Getting Started: Constructing Control Charts with Limited Data

> *It is very important to note that the answer to the question of how many measurements is in each case limited by the assumption that the variable X is controlled.*
>
> *Walter A. Shewhart, 1931*

> *Experience and theory both indicate that a subsample size of four is effective in the majority of instances that have come to my attention.*
>
> *Walter A. Shewhart, 1931*

We have seen that among the factors used to calculate control limits, the number of subgroups *k* and the subgroup size *n* play an important role. The amount of data you collect often determines both the number of subgroups that are possible and the subgroup sizes, so you might ask, "How much is enough?" That is, how much data do you need to have before you can begin computing control limits? For those who are comfortable with answers couched in terms of theory (e.g., coefficients of variation, degrees of freedom, and power curves showing the effects of the number and size of subgroups on the quality of the control limits), we direct you to Wheeler's advanced text [Wheeler 95]. Other interesting discussions

---

[35]The Central Limit Theorem says that, under very general conditions, when variation results from the sum of independent random elements and no small set of elements predominates, the distribution of the sum will approach a normal distribution as the number of terms in the sum increases. The convergence is often quite rapid. Thus, averages (e.g., $\overline{X}$) tend to be normally distributed, regardless of the underlying distribution. Ranges, though, always have distinctly nonnormal (skewed) distributions.

can be found in papers by Proschan and Savage, Hillier, and Quesenberry [Proschan 60, Hillier 69, Quesenberry 93].[36] The following paragraphs offer some advice that is based on the observations found in these references and elsewhere.

When calculating control limits for testing the stability of processes, it is *desirable* to base the calculations on a minimum of 25 to 30 subgroups if you are using X-bar and R charts, or 40 to 45 individual values if you are using XmR charts.[37] The reason for this is that a large number of subgroups reduces the influence that a few extreme values can have on the calculated limits.

Note the emphasis on the word *desirable.* It is not mandatory to have the suggested minimum number of subgroups, especially when getting started. It usually pays to construct a run chart and begin plotting tentative, trial limits as soon as possible. You must be cautious in interpreting the initial results, though, as the limits you compute with small amounts of data may not be especially reliable.

Shewhart offers this advice with respect to getting started [Shewhart 31]:

> How many subgroups of size four must we have before we are justified in using Criterion I [i.e., 3-sigma limits]? That this question is important is at once apparent because the expected probability of a statistic falling within the ranges established by Criterion I approaches the economic limiting value only as the total number n of observations approaches infinity. This difference in expected probability, however, even for two subsamples of four is likely less than 0.02 and certainly less than 0.05. Hence, the effect in the long run of using Criterion I when the total number of observations is small is to indicate lack of control falsely on an average of perhaps 5 times in 100 trials instead of 3 times in, let us say, 1,000 trials which it would do when the total number n is large. In almost every instance we can well afford to take this added precaution against overlooking trouble when the total number of observations is small. It appears reasonable, therefore, that the criterion [of 3-sigma limits] may be used even when we have only two subsamples of size not less than four.

If you use only a few subgroups when you plot charts for averages and range, it is possible that an unusually large range within one subgroup can increase the distance between the center line and the control limits on the chart for averages, thereby increasing the risk that you will miss an out-of-control signal. Keep in mind, though, that even when control limits are inflated, values that fall outside the limits remain probable out-of-control signals. Postponing the use of available data until you have collected 25 or more subgroups may cause you to miss opportunities for early discovery of out-of-control conditions.

---

[36] All of these analyses assume that the process is in statistical control.

[37] The Western Electric handbook is somewhat more liberal in its advice for starting XmR charts. It says, "Have 20 or more numbers if possible, but not less than 10 numbers [Western Electric 58]."

The advantage of early trial limits is that even when you cannot yet demonstrate stability, you can get early indications that assignable causes of variation are present. Concluding that a process is stable or nearly so, though, is risky with less than 20 to 30 subgroups. Often even more observations are advisable.

The subgroup size *n* also affects the computation of control limits. As the subgroup size increases, you will need more observations to determine whether or not a process is in control.

Increasing the subgroup size makes the control limits more sensitive to small shifts in the process [Montgomery 96]. While this may be desirable in some situations, it is usually more important to minimize the amount of variability within the subgroups. This is the principle of homogeneously subgrouped data that is so important to rational subgrouping: when we want to estimate natural process variability, we try to group the data so that assignable causes are more likely to occur between subgroups than within them. Control limits become wider and control charts less sensitive to assignable causes when large subgroup sizes lead to nonhomogeneous data. Creating rational subgroups that minimize variation within groups always takes precedence over issues of subgroup size.[38]

Our advice then is the same as that given by Shewhart, Wheeler, Montgomery, and others—when limited amounts of data are available, calculate and plot the limits, and seek out the assignable causes that the data suggest. Then update the control limits as more data become available.

Revising and Updating Control Limits

> *...companies must learn not to just accept a stable or controlled process. They must strive to improve their processes. Improvement may cause out-of-control conditions on the "good" side. If the improvement rate is slow, simply recalculate the control limits periodically. If the improvement is rapid, use run charts until the improvement rate flattens and process stability is again achieved.*
>
> *Beauregard, Mikulak, and Olson, 1992*

Revising and updating control charts both involve recalculating the control limits, but for different reasons and sometimes in different ways.

Revising. When you revise control chart limits, you are generally working with an initial set of data for which you have calculated tentative control limits, only to find that one or more data points fall outside the computed limits. If you believe that the out-of-limit values have

[38]We have already seen that it is possible to use a subgroup size as small as *n*=1 (XmR charts).

assignable causes, you may want to set these points aside and compute new limits that are not contaminated by the out-of-limit values. The new limits will be closer to the center line, which may result in additional points being outside the limits. This process of removing points and recalculating can be repeated until all points without known assignable causes are within the recalculated limits.

With X-bar and R charts, subgroups with out-of-limit ranges are the most likely candidates for removal. When control limits are revised by omitting certain subgroups or values from the computation of the limits, we do not delete the out-of-limit subgroups or points from the chart, but merely omit their values when calculating the limits.

There is an argument that says rather than revise the limits, go to work on removing the assignable causes, since you will have to collect new data and compute new limits after the assignable causes are removed, in any case. This is food for thought.

Montgomery offers this additional advice [Montgomery 96]:

> Occasionally, when the initial sample values of $\overline{X}$ and R are plotted against the trial control limits, many points will plot out of control. Clearly, if we arbitrarily drop the out-of-control points, we will have an unsatisfactory situation, as few data will remain with which we can recompute reliable control limits. We also suspect that this approach would ignore much useful information in the data. On the other hand, searching for an assignable cause for *each* out-of-control point is unlikely to be successful. We have found that when many of the initial samples plot out of control against the trial limits, it is better to concentrate on the *pattern* formed by these points. Such a pattern will almost always exist. Usually, the assignable cause associated with the pattern of out-of-control points is fairly easy to identify. Removal of this process problem usually results in a major process improvement.

Updating. We talk about *updating* control chart limits when we use additional, more recently collected data to recompute the limits for an ongoing chart. The need to update can occur if you began the control chart with less data that you would have liked (see the previous section), if the process is observed to have shifted, or if a deliberate change has been made to the process. In these cases, you should recalculate the limits by using the newly collected data plus any measurements obtained previously that remain applicable. When the process has shifted, though, or when you have made a deliberate change to the process, previously collected data may no longer be applicable.

Testing for and Sustaining Statistical Control

> *It has been observed that a person would seldom if ever be justified in concluding that a state of statistical control of a given repetitive operation or production process had been reached until he had obtained, under presumably the same essential conditions, a sequence of not less than twenty-five samples of four that satisfied Criterion 1* [i.e., fall within 3-sigma limits].
>
> *Walter A. Shewhart, 1939*

> *The fact that an observed set of values for fraction defective indicates the product to have been controlled up to the present does not prove that we can predict the future course of this phenomenon. We always have to say that this can be done provided the same essential conditions are maintained, and, of course, we never know whether or not they are maintained unless we continue the experiment.*
>
> *Walter A. Shewhart, 1931*

> *Statistical control is ephemeral; there must be a running record for judging whether the state of statistical control still exists.*
>
> *W. Edwards Deming, 1986b*

Although it is possible (and practical) to establish control limits, at least on a trial basis, with relatively small amounts of data, you should be cautious about concluding that any process is in statistical control. States of statistical control are ideal states, and testing for them is much like testing for error-free software. You may be able to show that you have not reached the state, but you can never prove or know for certain when you have arrived.

Nevertheless, the data you have, when plotted on control charts, are infinitely better than no data at all. Nothing can change the fact that you will have to bet on your predictions of the future performance of your processes. If your control charts signal out-of-control histories, you have little basis for extrapolating historical performance to the future. But as you identify and permanently eliminate assignable causes of unusual variation, your willingness to rely on extrapolation increases. Reliable predictions of process performance and the setting of achievable goals then become reasonable possibilities.

One important corollary to your caution in concluding that a process is stable is that you should almost never stop control charting any process, especially one that has had a history of going out of control. How else can you be assured that, once stabilized and made predictable, the process has not fallen back into its old ways?

7.5 The Problem of Insufficient Granularity in Recorded Values

When measured values of continuous variables have insufficient granularity (i.e., are coarse and imprecise), the discreteness that results can mask the underlying process variation. Computations for $\overline{X}$ and sigma can then be affected, and individual values that are rounded or truncated in the direction of the nearest control limit can easily give false out-of-control signals.

There are four main causes of coarse data: inadequate measurement instruments, imprecise reading of the instruments, rounding, and taking measurements at intervals that are too short to permit detectable variation to occur. When measurements are not obtained and recorded with sufficient precision to describe the underlying variability, digits that contain useful information will be lost. If the truncation or rounding reduces the precision in recorded results to only one or two digits that change, the running record of measured values will show only a few levels of possible outcomes. Fortunately, when this problem occurs, it is easy to identify.

Figure 7-2 shows two sets of values for *X* (the measured process performance) and *mR* (the moving range of *X*). The left-most set of values lists 32 observations as they were recorded; the right-most set lists the observations after rounding (or as they might have been recorded if the measurements were insufficiently precise). The XmR charts produced from the two sets of values are shown in Figures 7-3 and 7-4. Notice that the charts do not appear to describe the same process. The out-of-control points in Figure 7-4 appear solely because the data do not correctly reflect the underlying process variation.

	Measured		Rounded	
Observation	X	mR	X	mR
1	1.08	—	1.1	—
2	1.09	.01	1.1	0
3	1.15	.06	1.2	0.1
4	1.07	.08	1.0	0.2
5	1.03	.04	1.0	0
6	1.08	.05	1.1	0.1
7	1.1	.02	1.1	0
8	1.04	.06	1.0	0.1
9	1.07	.03	1.1	0.1
10	1.1	.03	1.1	0
11	1.12	.02	1.1	0
12	1.09	.03	1.1	0
13	1.03	.06	1.0	0.1
14	1.03	.0	1.0	0
15	1.09	.06	1.1	0.1
16	1.13	.04	1.1	0
17	1.02	.11	1.0	0.1
18	1.04	.02	1.0	0
19	1.03	.01	1.0	0
20	1.04	.01	1.0	0
21	1.14	.1	1.1	0.1
22	1.07	.07	1.1	0
23	1.08	.01	1.1	0
24	1.13	.05	1.2	0.1
25	1.08	.05	1.1	0.1
26	1.03	.05	1.0	0.1
27	1.02	.01	1.0	0
28	1.04	.02	1.0	0
29	1.03	.01	1.0	0
30	1.06	.03	1.1	0.1
31	1.02	.04	1.0	0.1
32	1.01	.01	1.0	0

Figure 7-2: Measured Values as Recorded and Subsequently Rounded

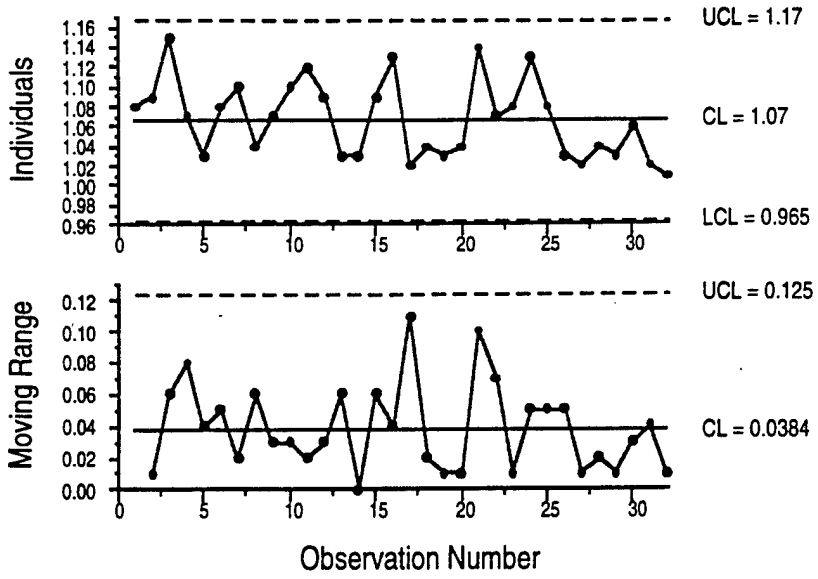


Figure 7-3: XmR Charts Constructed from Values as Originally Recorded

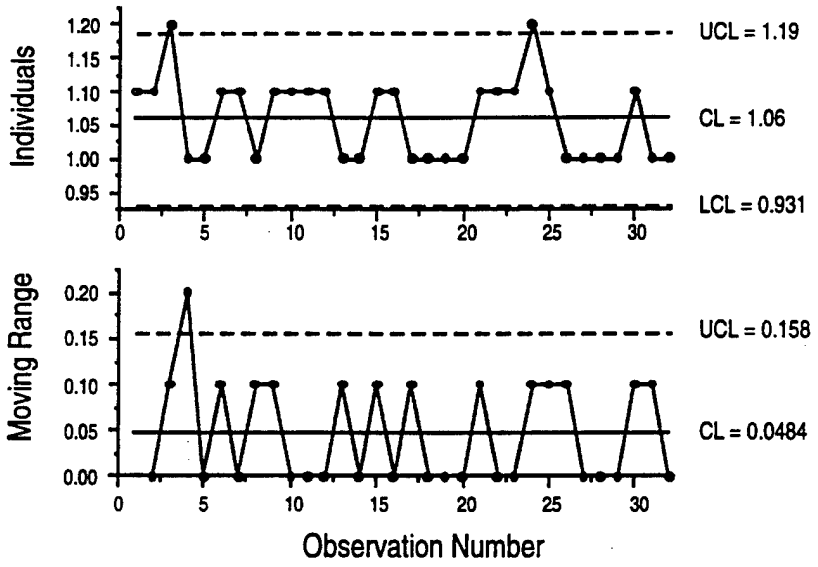


Figure 7-4: XmR Charts Constructed from Rounded Values

The solution to the problem of insufficient granularity is to ensure that the data used for control charts have a resolution that is smaller than the process standard deviation. A good rule of thumb for achieving this is to ensure that the set of points that are plotted always take on values that range over at least five possible levels of discreteness (the charts in Figure 7-4, for example, have stepped appearances because the values in them have only three possible levels). Increasing the number of levels can be accomplished by measuring more

precisely, by decreasing the frequency of measurement to allow for variation to occur between measured values, or by increasing the size of subgroups to allow for more variation within a subgroup.

Never round data to the point where the values that result span less than five attainable levels. If this rule must be violated, the data can be plotted in a running record, but they should not be used to calculate control limits.

Additional examples and guidelines related to this subject can be found under the topic of "Inadequate Measurement Units" in Wheeler's books [Wheeler 89, 92, 95].

7.6 Rational Sampling and Rational Subgrouping

> *...it is important to divide all data into rational subgroups in the sense that the data belonging to a group are supposed to have come from a constant system of chance causes.*
>
> *Walter A. Shewhart, 1931*

> *...emphasis...must be laid upon breaking up the original sequence into subgroups of comparatively small size. If this is not done, the presence of assignable causes will very often be overlooked.*
>
> *Walter A. Shewhart, 1931*

> *Obviously, the ultimate object is not only to detect trouble but also to find it, and such discovery naturally involves classification. The engineer who is successful in dividing his data initially into rational subgroups based upon rational hypotheses is therefore inherently better off in the long run than the one who is not thus successful.*
>
> *Walter A. Shewhart, 1931*

Control charts are founded on the concepts of rational sampling and rational subgrouping. These concepts deal with collecting and organizing data so that the questions at issue can be answered as reliably and economically as possible.

Rational Sampling

The purpose of rational sampling is to obtain and use data that are representative of the performance of the process with respect to the issues being studied. Rational sampling is concerned with the what, where, when, how, and why of measurements that are plotted and

used to compute control limits. The context in which the data are collected will always be the largest factor in determining the best ways to analyze the data.

Rational sampling is not limited to just collecting data. It also involves understanding the events associated with the data that have been collected. This understanding will guide you when selecting the observations and subgroups that you will use for computing control limits.

For example, when measuring items drawn from a production stream, the items should be selected in such a way as to preserve the information they contain about the process. If, for instance, outflows of parallel operations are combined, you will not be able to identify which of the operations gave rise to a result seen in a sampled item. When identifying the operation that caused the result is important, as it is when looking for assignable causes, this kind of sampling is not rational.

Another example occurs when a subset of the data is known to be unrepresentative of the process. Rational sampling says that the unrepresentative subset should be removed from the computation of control limits. This point can be controversial. The key to resolving the controversy lies in the word "unrepresentative." It is certainly wrong to omit a value or set of values just because you don't like what you see. There must be a stronger (rational) reason, perhaps of the sort, "Oh, yes—on that occasion X (not a normal part of the process) happened. We know that is an assignable cause, and we want control limits that will detect any similar occurrences." As we pointed out when discussing analytic studies, this is the kind of decision that only a subject-matter expert can make. Neither analysts by themselves nor any statistical tool can ever tell whether or not a particular set of data should be excluded from a study.

A third example of rational sampling deals with sampling frequency and its relation to the rates with which performance can change. Observations that are too far apart in time or space can permit substantial shifts, drifts, or changes in variability to occur between observations. When this happens, the variability within groups that is used to compute control limits will be too large, regardless of how the data are grouped. The process may then appear to be in control even though it is not.

Rational Subgrouping

Rational subgrouping is concerned with organizing the data so that the control charts answer the right questions.

The first requirement for rational subgroups is that, insofar as possible, subgroups should be selected so that the variations within any given subgroup all come from the same system of chance causes. This means that, in a practical sense, rational subgroups should consist of measured values from small regions of time or space. The purpose of small regions is to have sets of data that, within themselves, are relatively homogeneous with respect to potential causes of variation.

Selecting homogeneous subgroups is important because it is the variation *within* the subgroups that is used to compute the control limits for the process. Minimizing this variation makes detecting changes between subgroups easier and more reliable.

For example, if you look at how control limits are derived from range data, you will see that the ranges within subgroups are used to compute an average range across all subgroups. This average range is then used to compute the upper and lower control limits. When subgroups have been chosen so that variation within groups is small, the control limits will be as close to the center line as you can make them. Changes in process performance, such as shifts and trends, will then be more easily detected.

In effect, the variation *within* subgroups puts limits on the amount of variation that can exist *between* subgroups for the process to be stable. The larger the variation within groups, the wider the limits and the less effective the control chart will be in detecting instabilities. Another way of thinking about this is to understand that control charts ask the following questions:

- Control charts for averages ask, "Do the subgroup averages vary more than they should based on the observed variation within subgroups?"
- Control charts for ranges ask, "Is the variation within subgroups consistent from subgroup to subgroup?"

In short, the ways in which data are collected and organized determine the behavior you see reflected on a control chart. Inappropriate groupings and the wide limits associated with them will affect your ability to detect patterns and out-of-limit conditions. Variation within subgroups is what determines the sensitivity of the control chart to variation between subgroups. One of your principal objectives when choosing subgroups is to make the opportunity for variation within groups as small as possible. This means paying attention to not only how observations are grouped, but also how and when the data are collected.

Thus, it is important to consider the potential sources of variation and organize your data into subgroups that will help you address the questions you have relative to stability, capability, and process improvement. In some cases, there will be a single, natural way to group the data. In other cases there may be several ways. When many ways are possible, you should select subgroups that will permit the questions of interest to be answered. The following example illustrates some of these issues.

Example

> *The control chart tests the arbitrary or nonrandom arrangements of points to determine whether they behave as if they were random. If the plotted points indicate nothing but randomness, this tends to show that the variable which formed the base of the arrangement is not a significant variable.*
>
> *On the other hand, if the points indicate that nonrandomness has entered the data, this tends to show that the variable on which the arrangement is based is actually a significant variable.*
>
> *Statistical Quality Control Handbook*
> *[Western Electric 58]*

The software organization of Alphabetronics, Inc. is developing a product consisting of four major components. There is a different design team for each major component. The leader of the design group has asked the four design teams to count the number of fanouts (calls to other modules) as the designs for each of their modules are completed. The leader suspects that the number of fanouts per module might be a useful factor for characterizing system complexity. In particular, she anticipates that there may be a relationship between high fanout counts and defects [Card 90]. As a result, she wants to keep the number of fanouts low. Also, there are new members in each of the design teams, and the group leader wants to be sure that the designers, as a group, are producing consistent designs relative to the overall system structure.

The fanout counts for the first 20 modules designed by each team are shown in Figure 7-5.[39] The counts

Module Sequence Number	Fanout Counts for Team A	B	C	D	X-bar	R
1	1	4	6	4	3.75	5
2	3	7	5	5	5.00	4
3	4	5	5	7	5.25	3
4	2	6	4	5	4.25	4
5	1	6	7	3	4.25	6
6	3	8	6	4	5.25	5
7	5	7	6	6	6.00	2
8	3	5	4	6	4.50	3
9	2	5	9	4	5.00	7
10	5	5	6	7	5.75	2
11	4	5	6	5	5.00	2
12	5	7	8	6	6.50	3
13	3	3	7	3	4.00	4
14	2	3	6	9	5.00	7
15	3	7	4	3	4.25	4
16	4	6	6	5	5.25	2
17	0	5	5	5	3.75	5
18	3	4	6	6	4.75	3
19	0	4	4	6	3.50	6
20	2	6	5	4	4.25	4
Grand Averages					4.76	4.05

Figure 7-5: Module Fanout Data

[39] This example and the data in it are based on a comparable example given by Wheeler [Wheeler 92].

were recorded in the sequence in which each team completed its designs. Knowing that the sequence is a production sequence is important. If the sequences were recorded in the order inspected or reported instead of the order produced, the conclusions about stability and assignable causes that could be derived from the data could easily be different and perhaps less useful. This is part of what we mean when we say that traceability of values to the events that produced them is essential to valid analyses.

The data in Figure 7-5 have two identifiable sources of variation. There is variation between teams, and there is variation over time in the number of fanouts produced by each team. The variation within each team over time is shown in the columns labeled A, B, C, and D. The design group leader must decide how to group the data so that the sources of variation can be isolated and characterized. For purposes of illustration, we will do this in three different ways.

Since the design group leader wants to determine if the overall frequency of fanouts has been consistent as the design has been progressing, the counts will first be organized into subgroups of size $n = 4$ consisting of one value for each design team (subgroup 1 contains the first module produced by each of the teams, subgroup 2 the second, and so on). The averages and ranges for these subgroups are shown in the two right-most columns of Figure 7-5.

The design group leader prepared the control charts from the data as follows:

The grand average is $\overline{\overline{X}} = 4.7625$

The average range is $\overline{R} = 4.05$

The subgroup size is $n = 4$

From the table in Figure 5-10 (for $n = 4$): $A_2 = 0.729$ and $D_4 = 2.282$. D_3 is undefined.

Using the formulas for X-bar and R charts:

$$UCL_{\overline{X}} = \overline{\overline{X}} + A_2\overline{R} = 4.763 + 0.729(4.05) = 7.715$$

$$CL_{\overline{X}} = \overline{\overline{X}} = 4.763$$

$$LCL_{\overline{X}} = \overline{\overline{X}} - A_2\overline{R} = 4.763 - 0.729(4.05) = 1.811$$

$$UCL_R = D_4\overline{R} = 2.282(4.05) = 9.24$$

$$CL_R = \overline{R} = 4.05$$

$$LCL_R = D_3\overline{R} = \text{undefined (does not exist for } n = 4)$$

The results are shown in Figure 7-6.

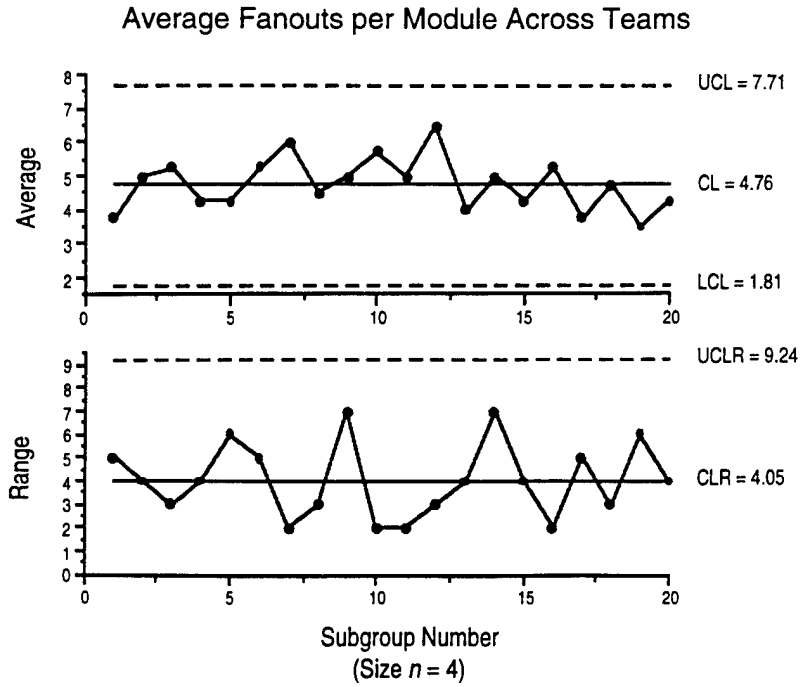


Figure 7-6: X-Bar and Range Charts for Module Fanout Data

The averages and ranges of the subgroups plotted in Figure 7-6 all fall within the control limits, thus meeting the principal requirement for statistical control.

The design group leader found this interesting, but she was concerned that the average fanout count was higher than desired. Also, the average range within subgroups was four, suggesting that there was considerable variability across the four teams.

To examine the difference in performance between teams, the group leader knew that she would have to organize the data so that each team became a subgroup. But this meant lumping together 20 observations for each team. This raised substantial questions in her mind about homogeneity and the rationality for such groupings. If the system of chance causes was not constant within each team, there would be no rational basis for inferring that the results of this kind of analysis would have any predictive value.

To explore the question of homogeneity within teams, the design group leader organized the data so that she could plot XmR charts for each team. This organization reduced the size of the subgroups to $n = 1$. The table that resulted is shown in Figure 7-7.

Module Sequence Number	Team Fanout Counts (X) and Moving Ranges (R)							
	A		B		C		D	
	X	R	X	R	X	R	X	R
1	1	—	4	—	6	—	4	—
2	3	2	7	3	5	1	5	1
3	4	1	5	2	5	0	7	2
4	2	2	6	1	4	1	5	2
5	1	1	6	0	7	3	3	2
6	3	2	8	2	6	1	4	1
7	5	2	7	1	6	0	6	2
8	3	2	5	2	4	2	6	0
9	2	2	5	0	9	5	4	2
10	5	3	5	0	6	3	7	3
11	4	1	5	0	6	0	5	2
12	5	1	7	2	8	2	6	1
13	3	2	3	4	7	1	3	3
14	2	1	3	0	6	1	9	6
15	3	1	7	4	4	2	3	6
16	4	1	6	1	6	2	5	2
17	0	4	5	1	5	1	5	0
18	3	3	4	1	6	1	6	1
19	0	3	4	1	4	2	6	0
20	2	2	6	2	5	1	4	2
Average	2.75	1.8	5.4	1.35	5.75	1.45	5.15	1.9

Figure 7-7: Data Organization of Module Fanout Counts for Constructing XmR Charts

The control charts that the group leader constructed are shown in Figure 7-8. The values for the teams are plotted on common coordinate systems to assist in comparing the results. The charts for individual values show no signs of out-of-control conditions for any of the teams. The range chart for Team C, however, shows one point that borders on its control limit. This point corresponds to the highest point (and biggest jump) on Team C's chart for individual values. Since this point is within its control limits and no other moving range appears unusual, the group leader saw no strong reason to say that Team C's performance was out of control.

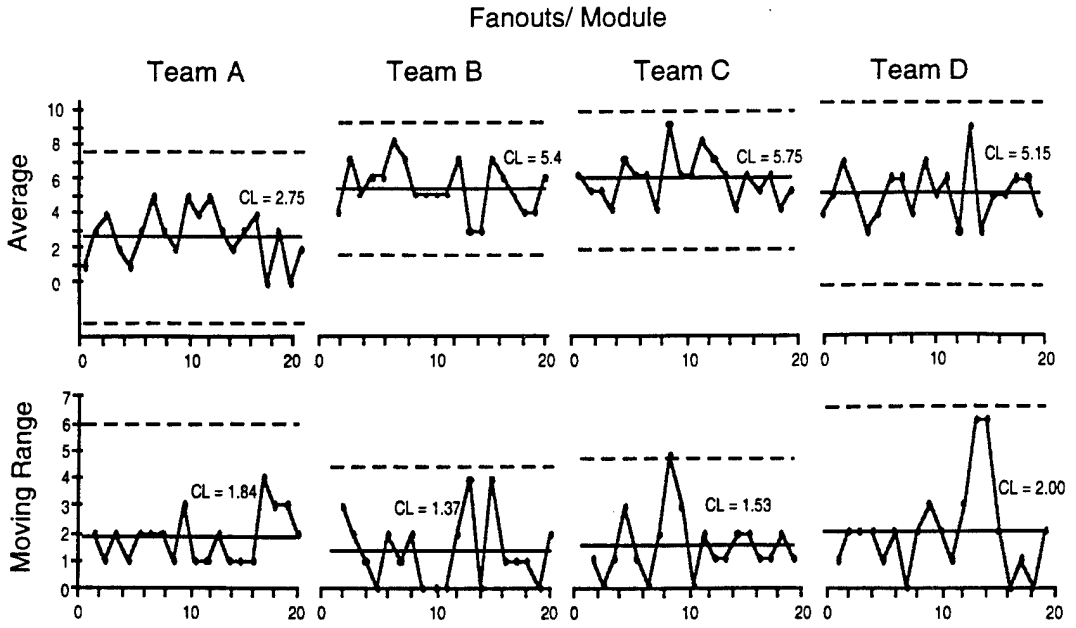


Figure 7-8: Individuals and Moving Range Charts for the Four Design Teams

The group leader was interested to see that Figure 7-8 showed the average fanout count for Team A's modules to be about half that of the other team's. This reinforced her suspicion that there were differences among the processes that the teams were using to design software modules. She knew, however, that the evidence so far was inconclusive.

Since the processes that the teams were using showed no evidence of trends and appeared to be in statistical control, the group leader felt that it would be justifiable to treat the data within each team as having come from a constant system of chance causes. This meant that she could combine the observations into four groups and proceed to the next step—comparing the performance of the teams.

To explore the question of variation in performance across teams, the group leader organized the data into four subgroups, as shown in Figure 7-9. Each subgroup is now of size $n = 20$, and the performance within the groups is known to be reasonably homogeneous.

The resulting control charts are shown in Figure 7-10. The average fanout count per module is the same as in Figure 7-6, but the limits are much closer to the center lines. This is not surprising, since we know that, in general, the standard deviation for the average of n values is proportional to

$$\frac{1}{\sqrt{n}}.$$

Module Sequence Number	Fanout Counts for Team A	B	C	D
1	1	4	6	4
2	3	7	5	5
3	4	5	5	7
4	2	6	4	5
5	1	6	7	3
6	3	8	6	4
7	5	7	6	6
8	3	5	4	6
9	2	5	9	4
10	5	5	6	7
11	4	5	6	5
12	5	7	8	6
13	3	3	7	3
14	2	3	6	9
15	3	7	4	3
16	4	6	6	5
17	0	5	5	5
18	3	4	6	6
19	0	4	4	6
20	2	6	5	4
X-Bar	2.75	5.4	5.75	5.15
R	5	5	5	6

Figure 7-9: Data Organization for Measuring Fanout Variation Between Design Teams

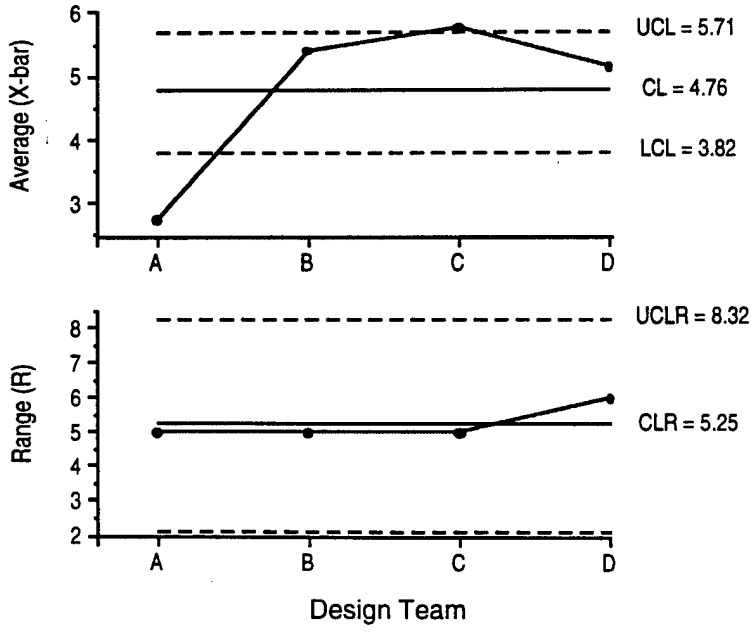


Figure 7-10: X-Bar and Range Charts for Fanout Variation Between Design Teams

Figure 7-10 shows that the average fanout counts for Teams A and C fall outside the control limits. This suggests that real differences exist between the teams, despite the performance of each team being under statistical control. The group leader would not be justified in reaching this conclusion had she not verified that each of the teams had a stable process.

Strictly speaking, because the sample size is bigger than 10, the group leader should have used an S chart rather than a range chart as the basis for assessing variability and computing control limits. Just to be safe, we verified her conclusions by making those computations. The results, as shown in Figure 7-11, are essentially the same.

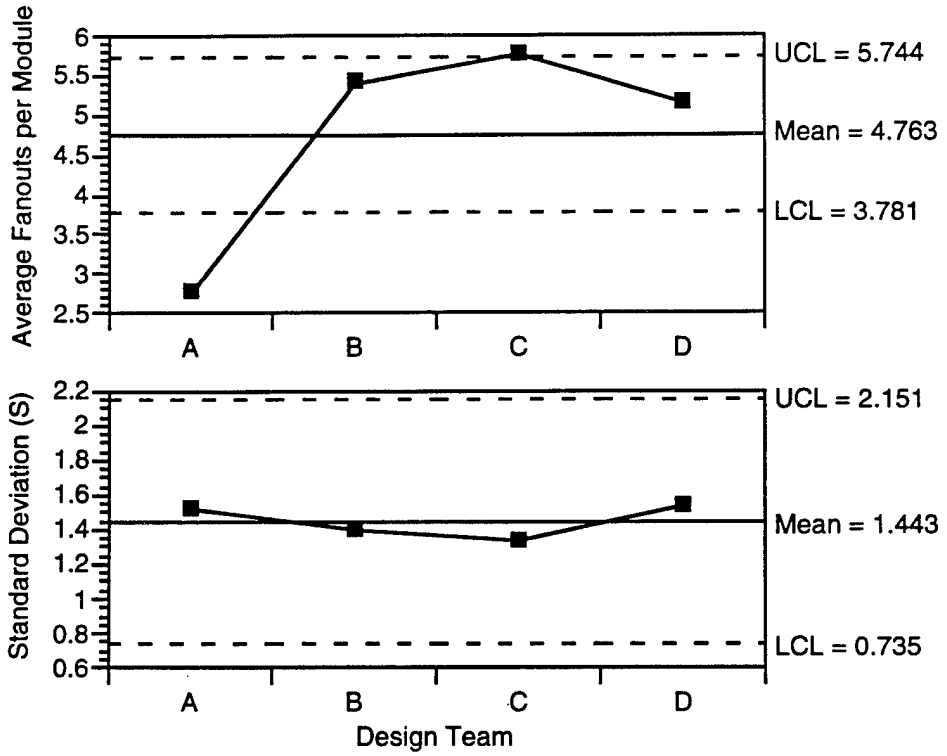


Figure 7-11: X-Bar and S Charts for Fanout Variation Between Design Teams

What is going on here? The range charts in Figures 7-6 and 7-10 provide a clue, but first let's review the questions the control charts are answering compared to the questions the design leader was asking.

The subgrouping of data shown in Figure 7-5 was aimed at examining the variation in fanout counts as the design progressed (i.e., over time). This subgrouping assumed that, at any point in the sequence, the variability across teams would be less than the variability over time. When this assumption is true, it is rational to use the variations observed among the four teams to compute control limits that describe the inherent variability in the average fanout count for groups of four at any point in time. These control limits can then be used to

examine the stability of the overall process. Thus, the charts resulting from the organization of data in Figure 7-5 examine these questions:

The X-bar chart:	"Have there been detectable differences in the average fanout counts as the design has progressed?"
The range chart:	"Has the team-to-team variability in fanout counts been consistent across the 20 groups of modules?

This organization of the data did not ask the question "Are there detectable differences in the fanout counts between the respective design teams?" In fact, there were implicit assumptions that the variation would be less across teams than it would be over time and that the performance of each team's design process would be evolving at the same rate.

The second organization of the data (Figure 7-7) made fewer assumptions. It addressed the problem of subgroup homogeneity by using XmR charts individually for each of the four teams. This data organization asked questions similar to the first organization, but aimed at the design teams individually. Because the subgroup size was $n = 1$, two-point moving ranges were used to estimate sigma and set the control limits. The questions that this organization of data asked were

The X-bar charts:	"Have there been detectable differences within any team's fanout counts as the design has progressed?"
The moving range charts:	"Has the variability in fanout counts for each of the teams been consistent across their 20 modules?

The third organization of the data used the fanout counts for each team's 20 modules as the subgroups ($n = 20$). The justification for doing this was that the XmR charts showed that each team's results appeared to come from a process with a constant system of chance causes. When the performance is consistent within teams (i.e., is in statistical control), the average of the variations in the teams (the subgroups) can be used to set limits for comparing the differences in performance among the teams. The questions being addressed with this data organization were

The X-bar chart:	"Were there detectable differences in the module fanout counts among the teams?
The range chart:	"Was the variability in fanout counts consistent from team to team?"

The X-bar chart in Figure 7-10 shows the average performance of each of the four design teams, and the range chart shows the respective ranges of performance within each team. Although the ranges are remarkably consistent across the teams, the chart for averages (X-bar) shows significant differences among the teams' average performance. When significant differences exist, there may be assignable causes. Whether or not Team A's process leads

to an unusually small number of fanouts or Team C's to an unusually large number are things to be investigated.

If the design group leader wants to obtain lower fanout counts from Teams B, C, and D, she must now determine the process changes that will be needed for the three teams to achieve results comparable to those of Team A.

Notice that sequence was not an issue in the third analysis. Thus, in a sense, this analysis does not illustrate a classical use of control charts. Instead, it effectively uses a technique called analysis of variance. The difference is that before the team leader performed her analysis, she verified that one of the necessary conditions—the existence of a single distribution within each team—appeared to hold. In ordinary analysis of variance, this is often just assumed. Now that you have seen how easy it is for data produced by processes to be corrupted by assignable causes, you can appreciate how easy it is for simple analyses of variance to lead you astray in dynamic environments like the ones that we often find in analytic studies.

The three approaches in this example show that there can be several ways to organize data into subgroups, and that each way makes different assumptions and leads to answering different questions. Clearly, the choice of subgroups determines what you can learn from the data. In general, the sources of variation of least concern (those closest to the inherent variation in the process) should be represented by differences within subgroups. The sources of variation that you wish to detect, on the other hand, should be represented by differences between the subgroups. Rational subgrouping keeps the variation within subgroups small, so that the tightest possible limits can be set. Narrow limits are what make control charts sensitive to the nonrandom variations you want to detect. Rational subgrouping enables you to set narrow limits without an uneconomical number of false alarms.

Thus, rational subgrouping is about common sense and efficiency in the use of data obtained from a process. The frequency of sampling should reflect both the nature of the process and the frequency with which actions or decisions are needed. If multiple measurements are collected at the same time and there are no structural reasons for the values to be different, they may be grouped together and used to estimate the inherent process variability. If only single values are collected, subgroups of size one and their associated moving ranges are likely to be most appropriate. When only limited amounts of data are available, you should look first to plotting a running record of the individual values. Automatically subgrouping limited amounts of data is not a good practice.

7.7 Aggregation and Decomposition of Process Performance Data

When analyzing process performance data, you must be constantly concerned that you have identified all sources of variation in the process. If a conscious effort is not made to account for the potential sources of variation, you may inadvertently hide or obscure variation that will help you improve the process. Even worse, you may mislead yourself and others with a faulty analysis.

When data are aggregated, you will be particularly susceptible to overlooked or hidden sources of variation. Overly aggregated data come about in many ways, but the most common causes are

- inadequately formulated operational definitions of product and process measures
- inadequate description and recording of context information
- lack of traceability from data back to the context from whence it originated

Overly aggregated data easily lead to

- working with data whose elements are combinations (mixtures) of values from nonhomogeneous sources
- difficulty in identifying instabilities in process performance
- difficulty in tracking instabilities to assignable causes
- using results from unstable processes to draw inferences or make predictions about capability or performance

We addressed issues associated with the causes and avoidance of overly aggregated data in Chapters 3 and 4. The example that follows shows some of the advantages that accrue when full traceability is present, so that data can be disaggregated and used to probe the underlying structure of a process. The example uses orthogonal attribute definitions to help identify sources of variation and provide insights to potential process improvement [Chillarege 92, 96].

Example

The top two rows in Figure 7-12 show the numbers of defects found during design inspections for 21 components of a new system. Each component is believed to possess the same area of opportunity in terms of the potential for defects to occur and be found.

XmR charts that were constructed for the total number of defects found in each component show no signs of instability from component to component (see Figure 7-13). Based on this

Component	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	Totals
Defects	12	16	18	32	22	16	23	35	15	27	16	25	20	26	20	23	23	36	22	27	17	471
Defect Type	Number of Defects per Type per Component																					
Function	3	5	4	4	4	3	3	20	4	11	2	3	3	5	3	7	4	5	5	15	2	115
Interface	2	2	4	4	3	4	2	3	3	4	2	3	5	3	3	3	2	16	6	2	4	80
Timing	1	1	0	1	1	0	2	1	0	0	2	0	1	1	1	1	1	0	1	0	0	15
Algorithm	0	0	1	14	2	0	0	0	0	0	0	1	5	2	7	6	5	1	2	0	1	47
Checking	1	1	5	1	7	1	1	2	0	1	6	3	1	12	1	0	2	4	3	5	2	59
Assignment	0	2	0	4	1	2	1	3	2	3	2	8	1	0	2	1	2	1	0	1	1	37
Build/Pkg.	3	1	1	2	1	0	0	4	3	6	1	0	2	1	1	1	3	2	2	2	1	37
Document	2	4	3	2	3	6	14	2	3	2	1	7	2	2	2	4	4	7	3	2	6	81

Figure 7-12: A Summary of Defect Types Found During Component Inspections

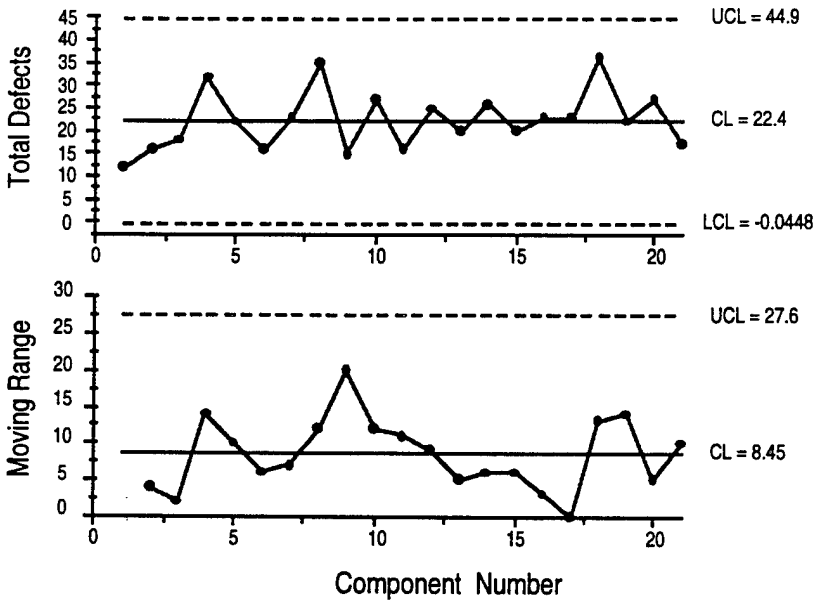


Figure 7-13: XmR Charts for the Total Number of Defects Found in Component Inspections

cursory examination, it would appear that the combined process of design and inspection was stable and under control.

The next activity that the organization had planned was to classify the defects according to the types listed in Figure 7-12 and determine if the number of defects found, by type, were within the target limits obtained from previous experience [Chillarege 92, Bhandari 93]. The numbers of defects of each type found in each component are shown in the bottom eight rows of Figure 7-12.

Before embarking on this activity, it is important to know whether or not the component design process is in control for the different types of defects found. Therefore, XmR charts were constructed by plotting the number of defects for each type in the sequence in which the components were completed. The control charts for these individual values are shown in Figure 7-14.

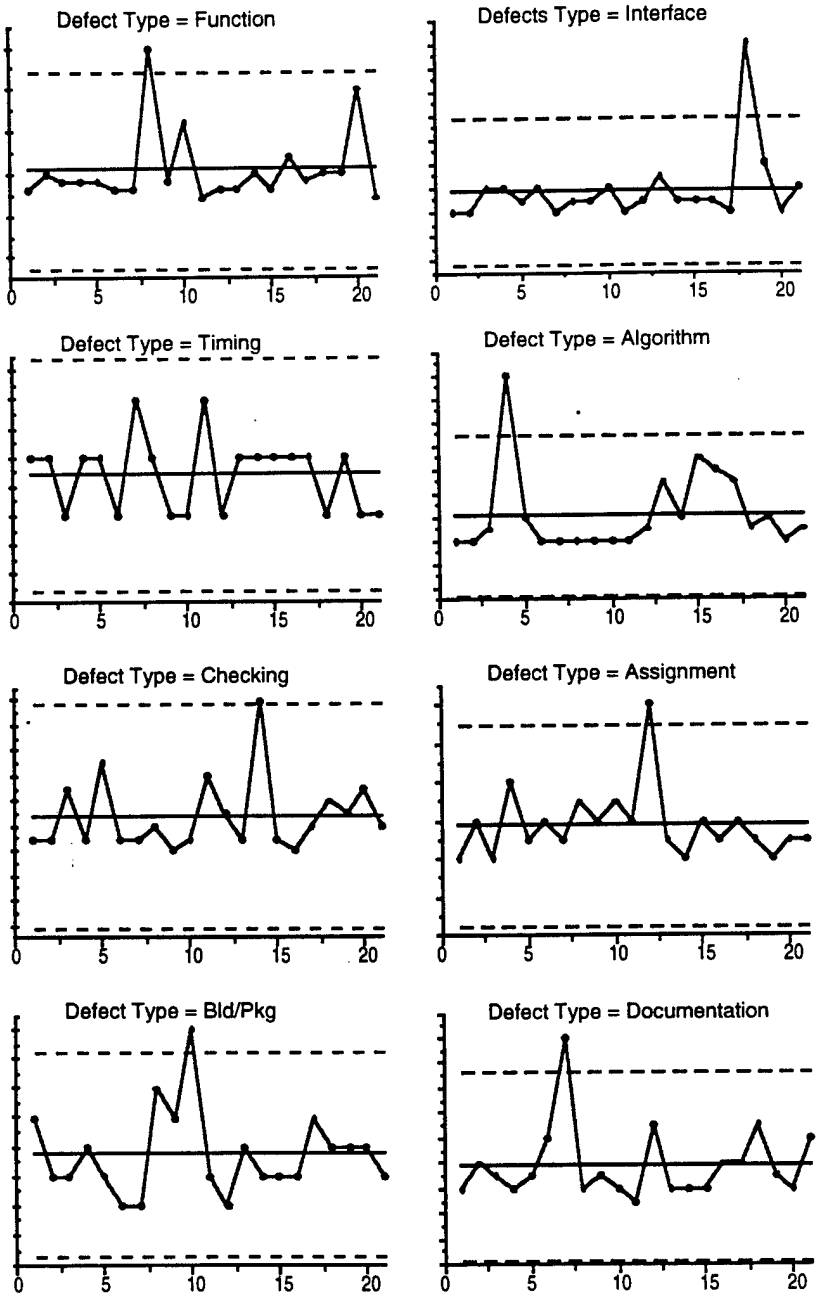


Figure 7-14: Control Charts for Individual Defect Types

Here we see that the disaggregated data for the numbers of defects found suggests unstable conditions in seven of the eight control charts. Several points are out of bounds, and one chart shows a run of eight points below the center line. (There were no signs of instability in the respective moving range charts.) The charts for individuals show that as many as eight different components may have been associated with the instabilities. This suggests that reasons for the unusual variations be sought. If reasons are found, actions can be taken to fix the problems that caused the unusual variations.

Note that when assignable causes are found and the corresponding points are removed from the charts, the control limits will become narrower, and the performance of the process will be more predictable for each defect type. The organization will then be in a position to make changes in a controlled fashion or to project future performance, if it desires.

This example shows that a single control chart of aggregated data, as in Figure 7-13, may be ineffective for identifying instabilities or for pointing to potential opportunities for improvement. As Wheeler and Chambers point out when discussing a similar example, the more sources of nonconformity that are combined on one chart, the greater the likelihood that the chart will appear to be in control [Wheeler 92]. Thus, although charts such as the one in Figure 7-13 may provide a useful record of what is happening, they are not very useful for improving a process. Due to the smoothing produced by aggregating the effects of several sources of error, control charts for the total number of defects can easily fail to identify both the timing of instabilities and the potential sources of assignable causes.

7.8 World-Class Quality

> *The failure to operate a process on target with minimum variance will inevitably result in dramatic increases in the Average Loss Per Unit of Production. Such losses may be severe, and they are always unnecessary.*
>
> *Conformance to Requirements, Zero Defects, Six-Sigma Quality, Cost of Quality and all other specification-based nostrums miss this point. World class quality has been defined by "on target with minimum variance" for the past thirty years! The sooner one wakes up to this fact of life, the sooner one can begin to compete.*
>
> *Donald J. Wheeler, 1995*

The most important use of a loss function is to help us to change from a world of specifications (meet specifications) to continual reduction of variation about the target through improvement of processes.

W. Edwards Deming, 1993

I know not why this works, other than the mathematics.

Genichi Taguchi (Apocryphal?)

In reading about the concept of process capability as we introduced it in Chapter 5, it would be easy to conclude, "Oh, that doesn't apply to me! I work on software, and we never have specification limits."

While it may be true that few software processes have explicit specification limits, it is always true that excessive variability and off-target performance are inefficient and costly. The man most responsible for bringing this point to the world's attention is Genichi Taguchi. The situation he pictured for the classical concept of process capability is illustrated in Figure 7-15. The figure shows the model implicitly used in industry when products are inspected and defective (nonconforming) items are discarded.

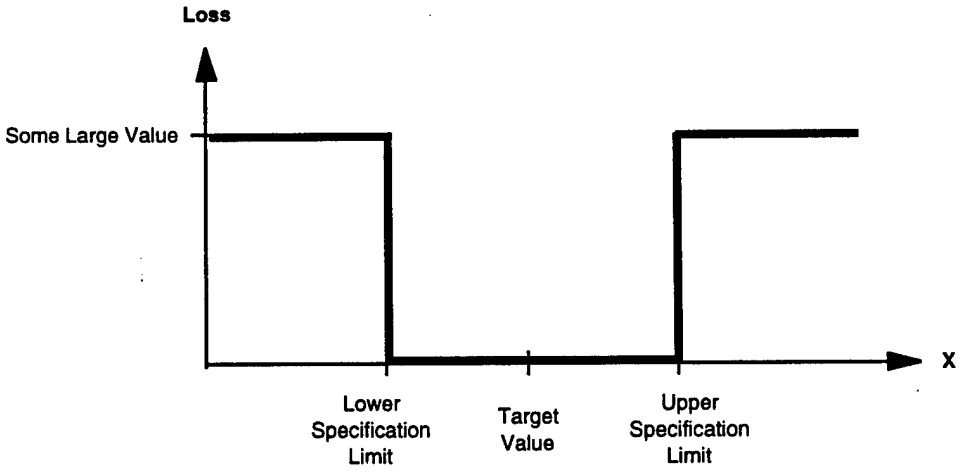


Figure 7-15: The Loss Function Associated With the Classical Concept of Process Capability

One of the characteristics that Taguchi observed in organizations that operate with this model is that processes that operate within specification limits get neglected. Taguchi recognized that this was inconsistent with the goal of continuous process improvement. He proposed an alternative model that recognized that all variation has economic

consequences, and that large variations are more detrimental than small variations. His model, in its simplest form, looks like Figure 7-16.

For both mathematical and practical reasons, Taguchi's model is often well approximated by a parabola, at least in the vicinity of the target value. This leads easily to analytic methods that, in conjunction with information about the distribution of outcomes, can be used to estimate the costs of being off target.

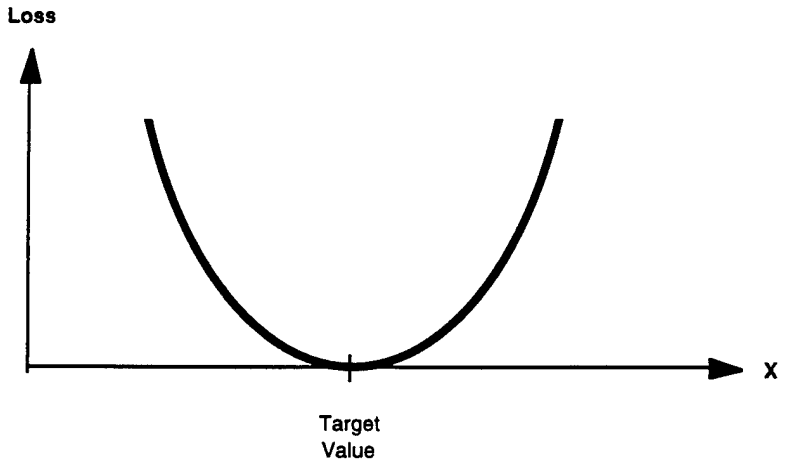


Figure 7-16: A Simple Loss Function Showing Taguchi's Concept of Capability

While the exact amounts of loss associated with being off target by given distances can be important, they need not concern us here. What is important, though, is to understand that any variability that exists causes deviations from the target, that each deviation has a cost, and that large deviations cost more than small deviations. All this is just as true in software settings as it is in manufacturing and service environments.

So, when you address issues of process performance that involve target values, do not conclude that the concept of capability does not apply just because no limits have been given to define an acceptable product or performance. Instead, we suggest that you picture Taguchi's model and adopt the view that

world-class quality = on target with minimum variability

When you adopt this view, the histograms you build with data from stable processes will give you much of the information that you need for assessing where you stand with respect to process quality and what the benefits would be from improving it. What's more, the attention of your organization will remain focused appropriately on opportunities for continuing to improve the process through additional reductions in variability.

8 Principles of Successful Process Measurement

This chapter lists some basic principles of software process measurement that we have assembled from the experiences, successes, and failures reported by numerous software managers and practitioners. These principles provide foundations for establishing guidelines and practices to be used when applying measures to software process management.

8.1 Process Measures Are Driven by Business Goals

Experience has taught us that we must identify the critical factors which determine whether or not we will be successful in meeting our goals. These critical factors often take the form of issues or problems, each representing a level of risk that jeopardizes our ability to meet our goals, responsibilities, or commitments. One way of identifying these issues is to pose questions relevant to the tasks or activities we must undertake to achieve our goals. The questions center around the entities that are involved in the tasks or activities that are performed. The nature of the questions depends on the tasks themselves, and our own experience and knowledge of tasks and activities provide the basis for the questions. The ability to pose questions that identify critical factors and issues is a function of what we understand or picture to be the tasks, resources, and products required to meet our goals. We refer to this picture as a mental model.

The knowledge and experience that you have acquired from working with a process and its constituent parts are the basis for such mental models. These mental models evolve and become fleshed out as you probe and make explicit your understanding of the processes you use and the questions you have about them. Your mental models are the engines that generate the insights that guide you toward useful measures and toward useful interpretations of measurement results.

Although mental models are abstractions, they are far from artificialities. We all use mental models every day to provide contexts and frameworks for translating observations into conclusions and actions. These personal mental models are seldom stated in explicit ways—but they always exist, if only in our minds. The models are derived from our personal experiences, and they exist even if we don't think consciously about them. They provide the context for interpreting and acting on the data we see in everyday life.

For example, Figure 8-1 illustrates one organization's perception of relations that exist between the introduction of software faults and where they are found. It shows an example of a mental model of a process that provides a basis for identifying measurable attributes.[40]

[40]This figure was given to us by Daniel J. Paulish of Siemens Corporation.

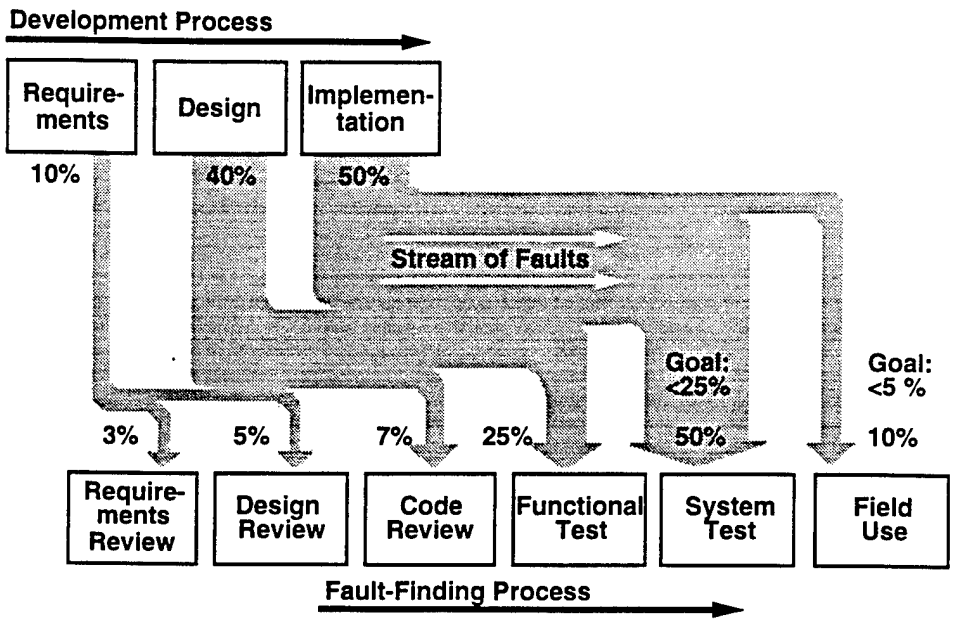


Figure 8-1: Process Models Help Us Construct Indicators—Fault-Stream Analysis

You may also have occasion to measure or characterize entities that lie outside your process models. For example, you should always keep in mind that your processes operate in environments that contribute to or detract from the processes' prospects for meeting your goals. Quantitative information about significant attributes of these environments will help you interpret information gathered from the processes that operate within them.

8.2 Process Measures Are Derived from the Software Process

A project development manager or process manager can measure only those products, processes, and resources that are being planned for or used by a project. Furthermore, the procedures used to capture data about these entities are largely determined by the tools, procedures, and methods that are in practice and available. In short, the developer's software process determines which attributes can be measured and how such measurements are to be made. Therefore, when establishing an overall measurement process, you must consider both the proposed and actual software processes of the developer.

It is also important to recognize that, in general, there may be significant differences between the software processes that are desired or envisioned and the ones that get documented, interpreted, or executed. These differences are frequently due to deficiencies in the description, comprehension, traceability, and interpretation of the process. There follows, therefore, a need to fully understand the software process (and possess an awareness of the factors that may affect the process) that is in direct proportion to the detail

of the measurement requirement. This understanding is required during the planning and analysis stages of the measurement process.

8.3 Effective Measurement Requires Clearly Stated Operational Definitions

> *Any physical measurement is the result of applying an operational procedure.*
>
> *W. Edwards Deming, 1986*

> *If you change the rule..., you come up with a new number.*
>
> *W. Edwards Deming, 1993*

A measurement process is based on collecting and analyzing well-defined data. All too often those who have the task of collecting measurement data or analyzing reported data are not given sufficiently complete specifications or descriptions of the data. As a consequence, assumptions are made that lead to incorrect collection or analyses.

Three criteria guide us in understanding the meaning of *well-defined* data [Park 92]:

- **communication**. Will the methods used to define measures or describe values allow others to know precisely what has been measured and what has been included in and excluded from aggregated results? Moreover, will every user of the data know how the data were collected, so that they can interpret the results correctly?
- **repeatability**. Would someone else be able to repeat the measurements and get the same results?
- **traceability**. Are the origins of the data identified in terms of time, sequence, activity, product, status, environment, measurement tools used, and collecting agent?

Note that these criteria are also crucial in supporting other measurement principles. Letting others know clearly what attributes have been measured and how the measurements were made is critical to both objective communication and traceability. Repeatability addresses the need for consistent interpretation over time and across all parts of the organization. Traceability supports the need for contextual information to guide interpretations and analyses.

8.4 Different People Have Differing Measurement Views and Needs

Different organizational entities (e.g., corporate, division, program, staff, project, or functional group) have different process management issues, perspectives, and interests, even when the same overall goals are shared among these entities. This often results in differing measurement needs and priorities. To the extent that these entities participate in the software measurement process, their measurement needs must be addressed.

In an operational sense, this means that each organizational entity must express its own measurement needs in terms of the attributes of the products, processes, and resources involved. Those responsible for planning and implementing a measurement process must be responsive to these needs while at the same time considering the capability and cost of collecting the requested data. The end result should be an integrated set of measurement requirements that balances the needs of the various organizational entities against the cost and capability of collecting the requested data.

8.5 Measurement Results Must Be Examined in the Context of the Processes and Environments That Produce Them

Analysis and interpretation of measurement data must be done within the context of other information about the process or product. Measurement data by themselves are neither bad news nor good news. A report indicating zero defects in the two months following product release may be very good news (if the product is being used by a large number of customers) or very bad news (if there are few to zero customers using the product). Measurement results must be examined in the context of other information about the product or process to determine whether action is required and what action to take. Unexpected measurement results generally require additional information to properly assess the meaning of the measurement.

8.6 Process Measurement Spans the Full Life Cycle

The measurement approaches and operational guidelines outlined in this guidebook apply throughout the entire software process. They also apply across projects and life cycles. Since process management and improvement are predicated on continual measurement of the process, it is important that all parts of the life-cycle process be subjected to process management scrutiny.

For example, activities associated with planning and estimating depend on the nature and proper execution of the planned software process. Inconsistent execution of the software process will lead to inconsistent and unreliable estimates for future projects. Measurements that help stabilize process performance will make estimates more reliable.

Conversely, estimates set the expectations that become embodied in the plan. Development activities are then measured against the plan and compared with process performance to determine if the process is under control and achieving its goals. Unstable estimating practices that are unreliable in predicting future performance may do more harm than good. The implication here is that estimating processes should be measured and examined for stability and improvement opportunities just like any other process.

Similar comments apply at every stage in the software life cycle. For example, tracking the process and product during post-deployment support periods often provides the data needed to improve those activities as well.

8.7 Retaining Data Provides Factual Baselines for Future Analyses

Historical records of process and product measurements must be retained so they will be available for identifying trends and establishing baselines for future planning and process improvements. When responsibilities for product planning, development, or post-deployment support change from one project to another, the sponsoring organization must assume the responsibility of establishing and sustaining a "corporate memory"—a repository containing pertinent project, process, resource, and product data. While this need not be (and probably should not be) a single, monolithic database, there should be a clear "chain of command" back to a person or office that is accountable for ensuring that a well-coordinated corporate memory is established and maintained.

8.8 Measures Are the Basis for Objective Communication

It is often said that a "picture is worth a thousand words." In measurement for software process management and improvement, this might be restated as "a valid measurement is worth a thousand guesses." Well-supported, objective numbers and facts drive out unsubstantiated, subjective opinions. Clearly understood factual data facilitate correct analysis and help ensure that there is agreement on what is happening as well as what should be happening in a software process. Software measurements can thus form the basis of clear, objective communication, not only within the software developing organization, but with customers as well.

8.9 Aggregating and Comparing Data Within and Across Projects Requires Care and Planning

Most data for software process management and improvement will come from individual software projects or the activities within them. These data will be specific to the particular software process in use, both in terms of their definition and the context in which the measured results are to be interpreted.

Since process management often involves collecting, analyzing, and comparing data from multiple projects and products, care must be used when aggregating or combining data as well as when comparing projects or products, plotting trends, or assessing the benefits of process improvements. When comparing or aggregating, the implicit assumption is that "all other things are equal." Differences among processes can easily invalidate an analysis or require changes in either the analysis or the way data are collected and aggregated. These problems can sometimes be avoided or alleviated by using data with common definitions from projects that used similar processes and measurement practices. This is not always possible, since the level of analysis may be more detailed than the level of commonality permits.

When comparable data are not obtainable directly, mathematical modeling can sometimes be used to normalize measured results. If models can be used to account for the factors that are different, it may then be valid to compare results obtained across projects and over time. The soundest mathematical models are those that have been validated by well-documented empirical evidence. The models must accommodate and account for the contextual information that describes the similarities and differences among the processes and environments where the data were collected. Thus, process management often requires collecting additional data beyond what is typically needed for project management. The purpose of this information is to enable valid normalizations, comparisons, aggregations, and analyses to be made.

8.10 Structured Measurement Processes Improve Data Reliability

Successful process management requires that measurements of processes, products, and resources be based on disciplined and controlled measurement processes. A successful measurement process includes not only clearly defined data, but also operational methodologies for relating process issues to the required measures and a defined process for collecting and analyzing measurement results. Making effective decisions to control, stabilize, predict, and improve software process performance requires confidence and assurance in the data. This means that measurement data must truly represent the process, relate to the issues at hand, and be analyzed with techniques that are consistent with the nature of the data.

9 Epilogue

We have used quotations throughout this guidebook to highlight important points and to give a sense of the importance that knowledgeable people attach to the concepts on which measurement for process management and improvement is founded. It seems fitting, then to close with a quotation like the one below. Nothing that we could write could better summarize the message of interplay between context knowledge and statistical methods that this guidebook has been trying to convey.

> *Every process and every product is maintained and improved by those who combine some underlying theory with some practical experience. More than that, they call upon an amazing backlog of ingenuity and know-how to amplify and support that theory. New-product ramrods are real "pioneers"; they also recognize the importance of their initiative and intuition and enjoy the dependence resting on their know-how. However, as scientific theory and background knowledge increase, dependence on native skill and initiative often decreases. An expert can determine just by listening that an automobile engine is in need of repair. Similarly, an experienced production man can often recognize a recurring malfunction by characteristic physical manifestations. However, problems become more complicated. Although familiarity with scientific advances will sometimes be all that is needed to solve even complicated problems—whether for maintenance or for improvement, many important changes and problems cannot be recognized by simple observation and initiative no matter how competent the scientist. It should be understood that no process is so simple that data from it will not give added insight into its behavior. The typical standard production process has unrecognized complex behaviors which can be thoroughly understood only by studying data from the product it produces. The "pioneer" who accepts and learns methods of scientific investigation to support technical advances in knowledge can be an exceptionally able citizen in an area of expertise. Methods in this book can be a boon to such pioneers in their old age.*
>
> *Ellis R. Ott and Edward G. Schilling, 1990*

Appendix A: Locating Key Information

This appendix provides references and pointers to help you locate key topics and examples.

Ideas and Examples for Measurable Entities and Attributes

Page Number	Figure Number	Figure Title
4	1-1	Business Goals, Project and Process Issues, and Related Measurable Attributes
26	2-6	Examples of Entities and Attributes that Can Be Measured to Address Process Compliance
31	2-12	Examples of Indicators for Business Operating Systems
31	2-13	Sources of Early Indicators for Customer Satisfaction, Flexibility, and Productivity
36	3-3	A Generic Process Model
36	3-4	A Simple Process Model for Defect Tracking
43	3-7	Measurable Entities in a Software Process
44	3-8	Measurable Attributes Associated with Software Process Entities
123	6-3	Compliance Issues and Potential Sources of Noncompliance
125	6-5	Inputs that Affect Process Performance

Pictorial Models of Process Management and Measurement		
Page Number	**Figure Number**	**Figure Title**
8	1-3	The Four Key Responsibilities of Process Management
11	1-4:	The Relationship Between Process Management and Project Management
12	1-5	Measurement Activities and Their Relationship to the Key Responsibilities of Process Management
33	3-1	Measurement Planning and the Key Responsibilities of Process Management
118	6-2	The Actions that Follow Evaluations of Process Performance

Procedures, Processes, and Approaches	
Page Number	**Topic**
20	Questions to answer if performance data are to be interpreted correctly
34–35	Steps for identifying process issues
40–44	Selecting entities and attributes for measurement
45	Templates for structuring measurement goals
52	Questions to address when analyzing existing measures and measurement practices
53–54	Things to do before writing a measurement plan
55	Checklist for preparing a measurement action plan

Procedures, Processes, and Approaches (continued)	
Page Number	**Topic**
59	Principal tasks associated with collecting and retaining data
63–65	Database planning issues
65–66	Database management issues
75	Control chart structure
79	Detecting instabilities and out-of-control situations (four tests)
81–83	The stability investigation process (10 steps)
85	Procedure for calculating control limits for X-bar and R charts
100	Conditions for Poisson distributions (six tests)
115	Procedure for assessing the capability of a stable process (flowchart)
118	The actions that follow evaluations of process performance (flowchart)
128–143	Analytic tools (examples)
144–145	Examples of improvement methods and technologies
148–149	Steps for collecting and analyzing measurement data
153	Additional steps for conducting an analytic study
200–202	Measurement planning template

Appendix B: Template for a Measurement Action Plan

This appendix provides a template for measurement action plans that is taken from *Goal-Driven Software Measurement—A Guidebook* [Park 96a]. You may find this template useful when identifying and structuring the key issues that you want your measurement plans to address. On the other hand, you (or your organization) may have alternative formats that you prefer. If so, by all means use any reasonable template that works well for you in your environment. But whatever your plan, be sure to address the issues that appear in this template. Each has been identified from experience accumulated in implementing action plans in real-world, people-centered environments.

Measurement Implementation Plan (a Template)

1. Objective

List the principal objective(s) of this measurement implementation effort. Identify the measures to be implemented, explain why they are important to your organization, and summarize the expected outcomes.

2. Description

Outline the origins of the plan, describe the goals and scope of the activities encompassed, and explain how the measures and efforts in the plan relate to other efforts and activities. The subsections that provide this information are described below.

Background

Give a brief history of the events that have led to or motivated this plan. Describe the origins of the plan, the work that has been done to date, who participated, and (optionally) the process that was used. Relate the planned actions to other existing or concurrent measurement activities within your organization and (if appropriate) in those of your customers or suppliers.

Goals

List and explain the goals that motivate and guide the activities under this plan. This section identifies three kinds of goals: (1) business goals, (2) measurement goals, and (3) the goals of this plan.

- The *business goals* frame the importance of the program and the level of support to be provided by senior executives.
- The *measurement goals* are more detailed and more specific. They guide the methods that will be used for collecting, storing, and using measured results. Each measurement goal should be identified and related to one or more of the business goals.
- The *goals for this plan* are more operationally oriented. They specify the outcomes that are sought. What do you want to achieve? How will you know when you are done? Often the most effective way to express these goals is in terms of exit criteria or criteria for success.

Scope

Relate the measures that this plan implements to the measurement goals they serve and describe their range of application. Do the measures apply to new projects only? To development projects? To procurement actions? To maintenance projects? To contractors and subcontractors? To large or small programs? To only certain divisions or departments?...etc.? Who are the major stakeholders? Who will be affected by the

measurement practices, processes, and methods? Who will use the results? Identify the time span over which this plan is to be effective.

Relationship to Other Software Process Improvement Efforts

Describe how the measurement efforts in this plan relate to other process improvement activities at your organization. Explain how the efforts relate to any goals or actions your organization may have established with respect to the CMM, the Baldrige Award, or ISO 9000 certification.

Relationship to Other Functional Activities

Describe how the measurement efforts in this plan relate to (and interface with) other functional groups and activities at your organization, such as cost estimating, time and effort reporting, cost accounting, procurement, technical writing, and quality assurance.

3. Implementation

Describe the actions that are to be taken to implement the measures identified in Section 2. For example, will you use pilot projects? Will you use focused subsets of the measures, perhaps locally, before broad organization-wide implementation? Put together a comprehensive strategy that addresses all aspects of implementation, including the tools and training needed to introduce, use, and sustain effective measurement. Address data-storage issues and the steps for incorporating these measures and measurement practices into your organization's policies, procedures, practices, and training curricula. Describe how you will use the measured results and how you will obtain feedback to continuously improve the measurement processes. Describe your plans for identifying problem areas and successes, and for publishing success stories and lessons learned. The subsections that provide this information are described below.

Activities, Products, and Tasks

Describe how the effort is to be accomplished. Partition the effort into manageable activities, products, and tasks that can be used as a basis for planning, reporting, management, and control. For each activity, product, or task, state the objective and identify the principal subtasks. Identify all sequences and dependencies that affect either the schedule or assignment of resources. Where possible, identify the entry and exit conditions that will determine start and completion of the task.

Schedule

Describe when each of the activities, products, or tasks is to be accomplished. Use Gantt charts, PERT charts, or alternative displays where appropriate to describe sequences and dependencies. Translate key actions, events, and deliverables into milestones so that performance can be tracked against plans.

Resources

Describe the resources that are being allocated to this effort. Address personnel, money, facilities, teaming plans, computer resources, and so forth.

Responsibilities

Name the individuals or groups that will be responsible for overseeing, planning, implementing, managing, approving, and funding this effort. Assign responsibility and authority for acquiring tools, for training, and for implementing and operating databases.

Measurement and Monitoring

Describe how the progress of implementing these measures will be measured, analyzed, and reported. Identify replanning points and describe how significant schedule deviations or changes and revised funding needs will be handled.

Assumptions

Identify the key assumptions upon which this plan is based. Key assumptions are ones which, if not satisfied, pose risks for successful implementation.

Risk management

Describe how you will identify, assess, track, and do contingency planning for the risk factors associated with the measurement implementation efforts covered by this plan. Describe the actions that will be taken to monitor the assumptions, and provide mechanisms for reacting if assumptions are not met. Also, identify all places where planned schedules and resources differ from estimates and describe the actions that are being taken to make the planned outcomes achievable.

4. Sustained Operation

Describe the actions that will be taken to sustain and use the measures implemented in Section 3. Assign resources and responsibilities and make provisions for continuing evolution. Describe the practices that will be used to evaluate and monitor the effectiveness of the measures and to assess their business value and their effects on organizational performance. Alternatively, if appropriate, provide direction and resources for preparing an operational plan for sustaining the collection, use, retention, evolution, and evaluation of these measures.

Appendix C: Example—A Practitioner's Report

This appendix provides a copy of a paper that was presented by Mark Roberts of McDonnell Douglas Corporation at the 1996 Software Engineering Process Group Conference in Atlantic City, New Jersey [Roberts 96]. The paper describes the experiences Mark and his organization encountered as they began applying control charts and statistical process control methods in software settings. The logical approach is instructive, as are the false starts and lessons learned. The paper is well worth reading. Understanding the issues that it illustrates will help speed your success in bringing stability and predictability to your own software processes. It offers a good example of using the principles of statistical process control to improve one's ability to separate authentic signal from random noise.

We thank Mark Roberts and the McDonnell Douglas Corporation for granting permission to reproduce the paper here.

Experiences in Analyzing Software Inspection Data

Mark A. Roberts
Software Engineering Process Group
McDonnell Douglas Aerospace
McDonnell Douglas Corporation
St. Louis, MO
mroberts1@mdc.com

May 21, 1996

C.1 Background

The software engineering community at McDonnell Douglas Aerospace (MDA) consists of over 1000 engineers working on a large variety of missile, aircraft, ground systems, and other support systems projects. In support of continuous improvement aimed at reducing costs and maintaining a competitive position, MDA management committed in 1991 to a software process improvement initiative based upon the Software Engineering Institute's Capability Maturity Model (SEI CMM). An SEI-licensed software process assessment conducted in early 1992 assessed a small community of avionics projects at capability maturity Level 2. Soon thereafter, MDA established an organizational goal to achieve maturity Level 3 in 1994. This goal was flowed down to our major aircraft and missile programs. Responsibility for establishing organizational processes and training to support our efforts to achieve Level 3 was assigned to our Software Engineering Process Group. The SEPG established an Organizational Standard Software Process (OSSP) to serve as the primary vehicle for process improvement. MDA's implementation of an OSSP was multi-faceted. Company-wide policies were written to address the commitment practices of the Capability Maturity Model (CMM). A standard life-cycle model and a set of process requirements were documented in the Software Engineering Process Manual (SEPM). Process and product metrics were documented in the Software Engineering Metrics Manual (SEMM). The SEPG also developed guidebooks to serve as primers for such processes as peer reviews, configuration management, and software development planning. One of the major sections in our guidebook was devoted to software peer reviews. The peer review guidebook contains the following sections:

- an overview of the MDA Software Peer Review Process
- definition of roles and responsibilities of participants
- a process flow and textual description of the actual peer review process, including inspections, walkthroughs, and colleague reviews
- a list and description of metrics to be collected, and
- guidelines on tailoring the process for specific projects.

A peer review training course was also developed by the SEPG to support dissemination of the process throughout the organization. The course consists of five hours of training on company time and includes an overview of the process followed by two lab sessions, where a facilitated inspection is conducted. The training is required for all MDA software engineers, software quality engineers, and software contractors. The peer review process has been used on many MDA software projects since early 1992. Our process required projects to collect summary data on defects found during peer reviews.

C.2 Problem

MDA software projects have been performing inspections and collecting defect data for several years. The peer review process was developed to support MDA's efforts to achieve SEI Level 3, so most of our projects included it as part of their defined software development process. The projects have been stockpiling the defect reports and have not used the data to (1) measure the effectiveness of the peer review process, or (2) indicate areas for improvement in the peer review process or the software development process. They have the data on paper, but don't know how to interpret the data and don't understand potential uses for the data. This situation causes people to question why they are collecting data in the first place. Our next challenge is that the SEPG has developed plans to move toward SEI Level 4. The findings from our Level 3 assessment indicated the need to look at the Quantitative Process Management KPA. Our plans relative to this KPA included training of the SEPG in statistical process control techniques and implementation of a comprehensive organizational metrics program. Our metrics team needed some "live" data so we could gain some experience in analyzing metrics. We also wanted to assess the benefits of the processes we implemented during our Level 3 activities to support our advancement to Level 4.

In 1995, MDA established a metrics team to support our SEI Level 4 efforts relative to organizational metrics. One of the first tasks of this team was to develop a survey for distribution to all MDA software personnel to help develop goals for metrics activities. Results of the survey indicated that the primary use of the metrics in the software engineering organization was progress monitoring. However, respondents indicated a desire to get more from metrics than simple software development progress or percent complete. Because the majority of the respondents were developers, these results indicated the need for metrics that benefit software developers directly. One of the common themes we discovered from the respondents was that the reasons behind collecting the metric data needed to be well understood. Many respondents recognized the benefits of collecting metrics but were not realizing them, while others wanted to see the benefits before "buying in" to their use. The metrics team then reviewed the metrics defined in our software metrics manual to determine the best starting point to help software engineers realize the benefits of collecting and analyzing metric data. Our metrics team identified peer review data analysis as a way to provide immediate metrics support to many of our software development projects.

C.3 Selecting a Set of Data

We began an initiative to develop a process to support analysis of inspection data during the software development cycle. Our team selected two projects to provide defect data for analysis. These projects had an existing history of defect data and were either in the latter stages of a software development effort or had just completed a software release. This paper describes our experiences in analyzing the data from the first of these two projects.

The project we identified as the subject of our analysis provided data from 212 inspections performed over a period of nine months. These inspections covered detailed design, code, and unit test plans. Figure C-1 shows the Inspection Report used to document summary information for each inspection, and Figure C-2 shows the Defect Report used to record each defect found during an inspection. The highlighted areas indicate the data provided to the metrics team.

INSPECTION REPORT

PROJECT	MAJOR RELEASE	PHASE	WORK PACKAGE	INSPECTION #
PRODUCT DESCRIPTION				
INSPECTION DATE	TIME	LOCATION	METHOD: ☐ MEETING	☐ ROUTING

		Prep. Time
Author		
Moderator		
Reader		
Tech Lead		
Tester		
Quality		
Traceability		
# of Inspectors		

Start Time:	
End Time:	
Total Group Insp. Time	
Total Preparation Time:	
Total Time Used:	

Reinspection Required?	
Est. Rework Hours:	
Est. Rework Due Date	
Actual Rework Hours:	
Actual Completion Date	

Distribution Date	
# of Units Inspected	
# of Pages Inspected	
# of Lines Inspected	
# SLOC Inspected	

Tech Lead Signoff	
Quality Signoff	

Figure C-1: Inspection Report

DEFECT REPORT										
PROJECT		MAJOR RELEASE		PHASE			WORK PACKAGE		INSPECTION #	
PRODUCT DESCRIPTION										
PRODUCT				DEFECT					REWORK	
Type	Page	Line	Unit Type	Code	Severity	Category	Origin Phase	Defect Description	Complete	Verified

Figure C-2: Defect Report

We worked independently from the project that provided us the defect data, with no insight into the product under development or their software development process. We only had the project's documented peer review process and the summary inspection reports - we did not have access to the detailed defect reports. One advantage of using the data provided was that we could be completely objective in our analysis. We did not know which engineers developed the information that was inspected or who participated in any of the peer reviews.

C.4 Data Analysis—Round 1

Microsoft Excel provided the capabilities we needed to reduce and analyze the data, so we loaded all of the data into Excel spreadsheets. The summary data included the Inspection ID, date, number of defects by defect code (or category) and size of the review package for each inspection. This project did not use source lines-of-code count for the code inspection - they had a program that counted a total "number of lines" for the entire review package. This count included textual paragraphs, code, and comments. Due to the unavailability of source lines-of-code (SLOC) counts, we selected "number of lines" as a means of normalizing the data from the inspections. We first developed a pie chart (Figure C-3) depicting the ratio of each defect category to the total number of defects. The pie chart indicated the defect categories that occurred most often in the inspections, but since the project used 15 defect categories, those categories with few total defects were lost in the noise. Normalizing the data using number of lines allowed us to develop an attribute control chart (a c chart) for the number of defects per line. Attribute control charts are used to determine if a process is "in a state of control". The c charts we developed (Figure C-4) looked reasonable - they showed that most of the inspections resulted in a normalized "number of defects" within the control

limits. But what did they mean? We decided we needed some help, so we enlisted the services of MDA's Variability Reduction (VR) group whose purpose is to provide training and facilitate the use of statistical tools to support process improvement throughout the organization. The VR group works primarily with production processes at MDA, so this study provided them an opportunity to apply statistical process control techniques to software.

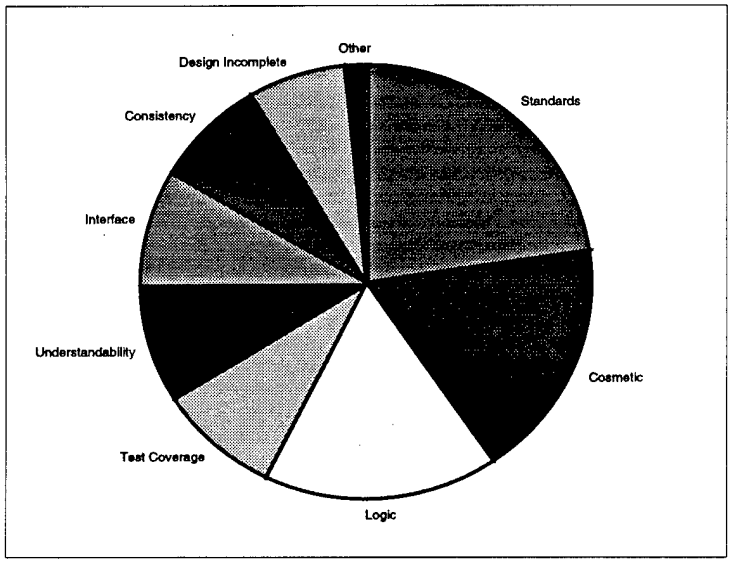


Figure C-3: Code Inspection Pie Chart

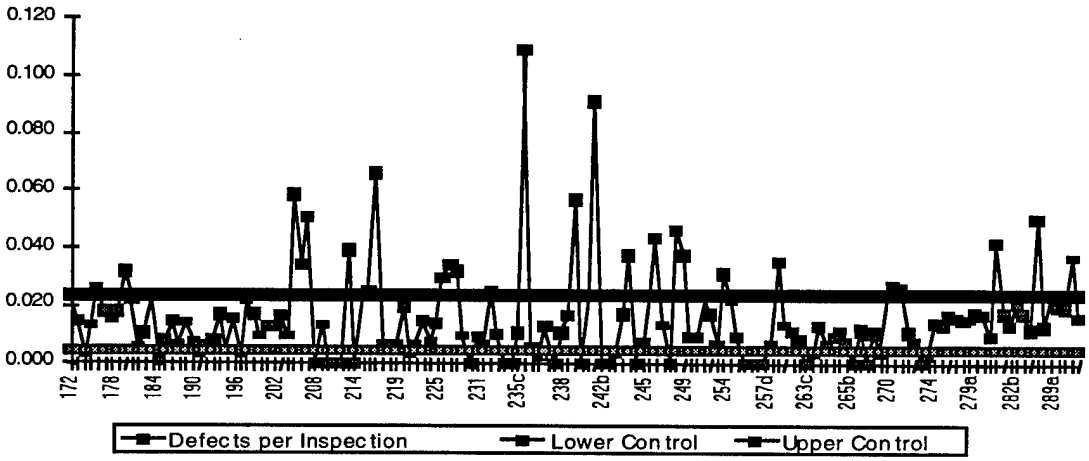


Figure C-4: Code Inspection C Chart

Our VR experts pointed out that c charts are only valid if the sample size is the same from inspection to inspection. This situation is very unlikely in software inspections, because it would require reviews to be conducted on the same size of review material, that is, always inspect the same number of SLOC or the same number of paragraphs, pages, etc. Therefore we needed to create u charts. U charts are another type of attribute control charts, but, unlike c charts, can be used when the sample size varies. The VR group also suggested that we step back and develop Pareto charts of the data to help determine how to proceed.

C.5 Data Analysis—Round 2

After the false start, we sat back to determine what we wanted to learn from the data. We also decided that we had too much data to handle and should break it down into more manageable pieces. The team decided to concentrate on the 124 code inspections and save the detailed design and test case reviews for later analysis. We chose code inspections as our first category because they are performed on more MDA projects, and they would provide more information on actual software implementation problems.

First, we wanted to identify the "heavy hitters" and/or the "vital few" defect categories. So we created a Pareto Chart (Figure C-5) and determined that of the 15 defect categories, the "heavy hitters" were Standards, Logic and Cosmetic. Of the 1896 defects found during code inspections 704, or 37% were categorized as Standards or Cosmetic defects. This data caused us to ask, "Why did they find so many standards errors?" A conference with the project's peer review process owner revealed that (1) they had A LOT of documentation and coding standards, (2) their customer agreed to this list of standards, and (3) many of the software developers were relatively new to the project and had not been fully trained in use of the standards. This Pareto chart did not, however, give us the "vital few." High quantity doesn't necessarily mean most critical; therefore, we needed to use severity codes to weight the defects.

Second, we wanted to answer the question, "Is the process in control?" In other words, "Is the development process consistent regardless of the size of the product being reviewed?" To help answer this question we created attribute control charts (u charts) for the top three defect categories. Upper and lower control limits (UCL and LCL) and centerlines (U bar) were calculated for all the code inspections which resulted in the plot shown in Figure C-6. In this figure the defect data is represented by (U).

From the u chart of logic defects we were able to determine that only six percent of the inspections found more logic errors than expected, indicating that, for the most part, the process is in control. We identified each inspection where the defects spiked above the UCL and provided the data to the project managers. To determine the root cause for those instances when the number of defects spiked above the UCL, project personnel reviewed the detailed inspection defect reports to look for changes or breakdowns in the development process, resources, personnel, etc., that could have caused the process to be out of control.

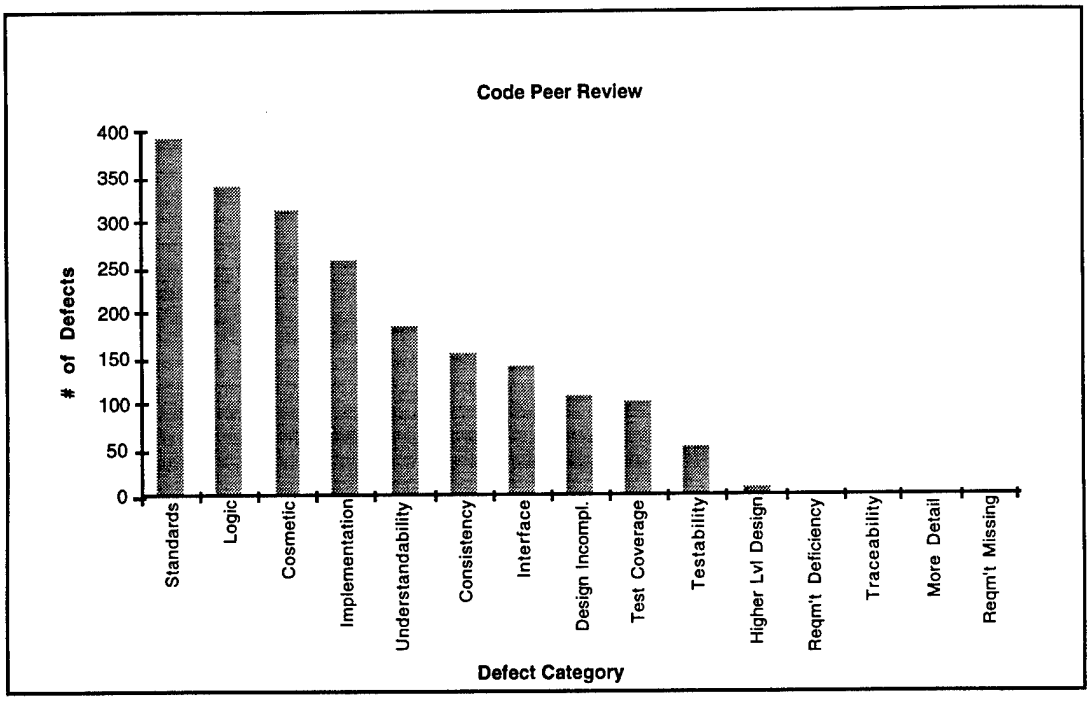


Figure C-5: Code Inspection Pareto Chart

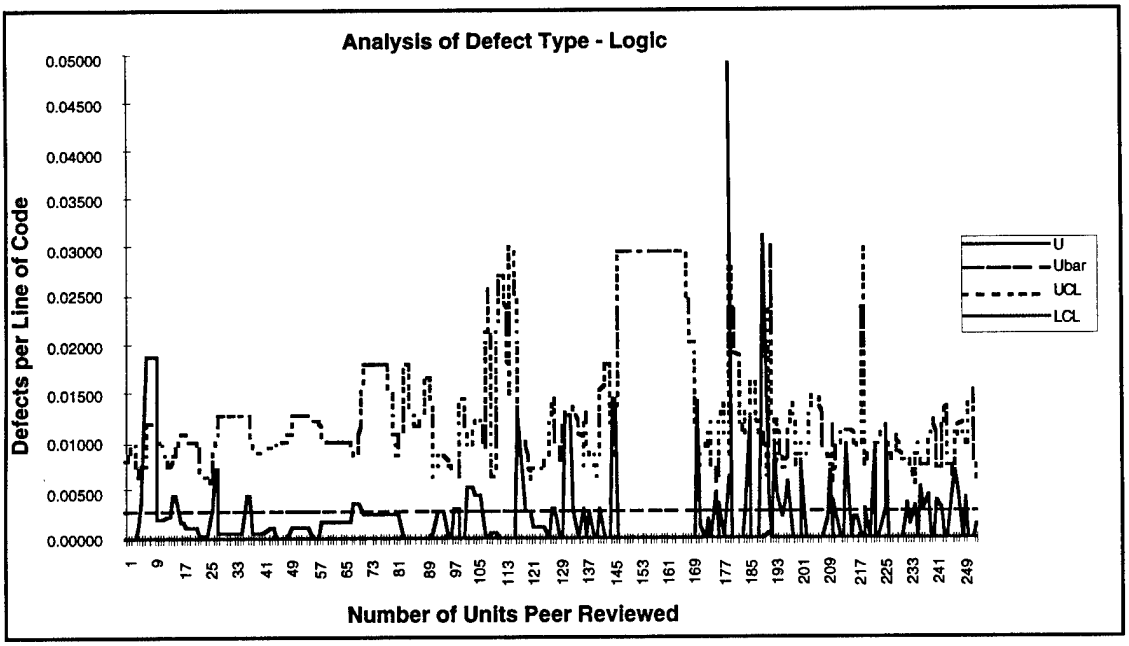


Figure C-6: Logic U Chart

C.6 Round 2 Wrap-Up

As a result of the second round of data analysis, we determined that the extraordinarily large number of standards and cosmetic defects were probably skewing our data and preventing us from concentrating on defects that may affect the functionality of the software. We obtained a coding standards checker and provided it to the project to help reduce the standards defects in the future. The project began an evaluation of tools such as "pretty printers" to aid in detecting or eliminating cosmetic errors prior to the peer review. Based on the data we provided to them, the project decided to reduce the number of defect categories by combining "understanding" with "more detail", "design incomplete" with "higher level design", and "requirements deficiency" with "requirements missing". Both the project and our metrics team felt that fewer numbers of defect categories will reduce future effort in analyzing the data, while still providing enough insight into the process.

C.7 Data Analysis—Round 3

For our next round of data analysis we decided to do some additional investigation of the current data based on what we learned from the previous activities. First, we combined the defect categories to match the process changes made by the program and then reran the Pareto charts. This produced very little change; however, the charts had a much cleaner look due to the reduced number of defect categories. Although the charts looked better, they didn't tell us anymore than the original charts.

To achieve our goal of determining the "vital few", we decided to weight the defects based on the minor and major severity classifications for the combined defect categories. In our initial analysis, we simply added the number of major and minor defects to determine a single count for each category. Next, we ran Pareto charts with major defects carrying three times the weight of minor defects. This weighting typically moved logic errors in front of the standards and cosmetic errors, which we expected to see; however, the Pareto charts still did not provide enough emphasis on the defects that affected the functionality of the software. We then ran the Pareto charts again - this time using a weighting factor of ten for the major defects. This made a significant difference. The resulting charts (Figure C-7) emphasized the defect categories that would be most important during each software development phase, moving defects like standards and cosmetic into the 4th and 5th positions. Using a weighting factor of ten we felt that we finally discovered our "vital few".

While we continued our analysis, research by the project process personnel of the inspections where the logic errors exceeded the UCL, showed the high number of defects could be attributed to special causes (for example, new software personnel and specialized software modules). Based on the theory of control charts, we eliminated those 12 inspections and recalculated the upper and lower control limits and the centerline for the remaining logic errors. This new u chart showed only two inspections with errors outside the control limits. Data from the original u chart is shown in Figure C-8. It has been plotted as a short-run attribute control chart, where UCL and LCL (+3.00 and -3.00) for each inspection

have been normalized to improve readability. Figure C-9 depicts the same data after elimination of the 12 inspections with "special causes".

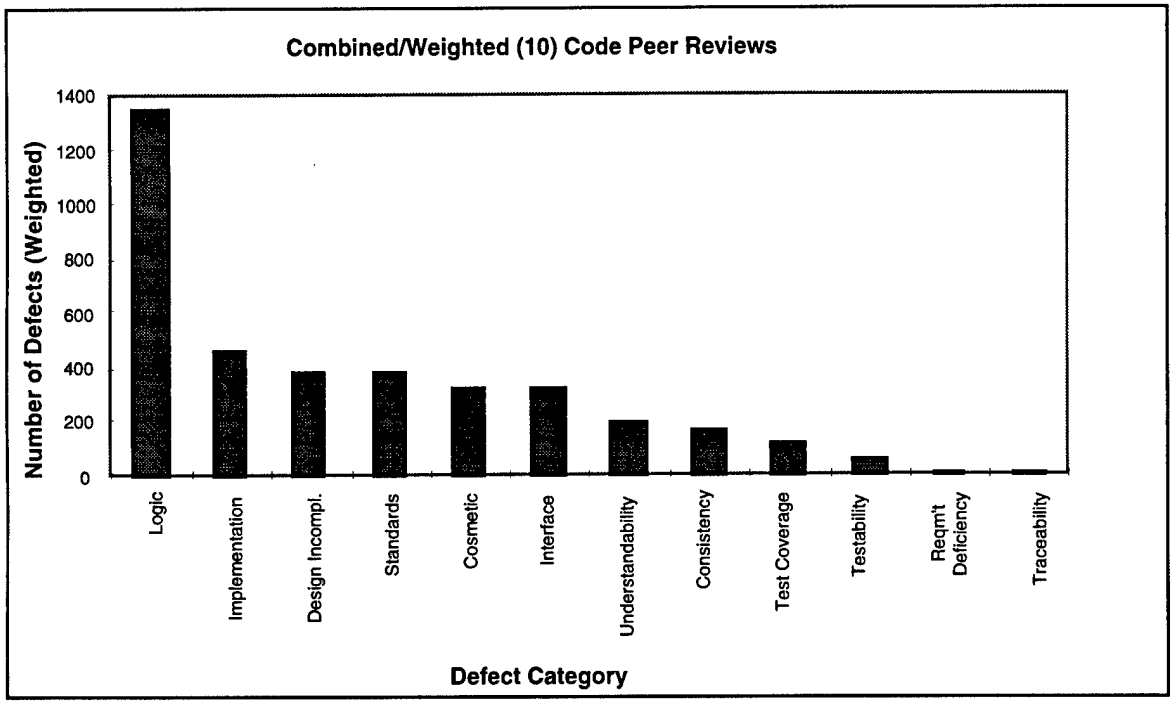


Figure C-7: Combined/Weighted (factor of 10) Code Inspections

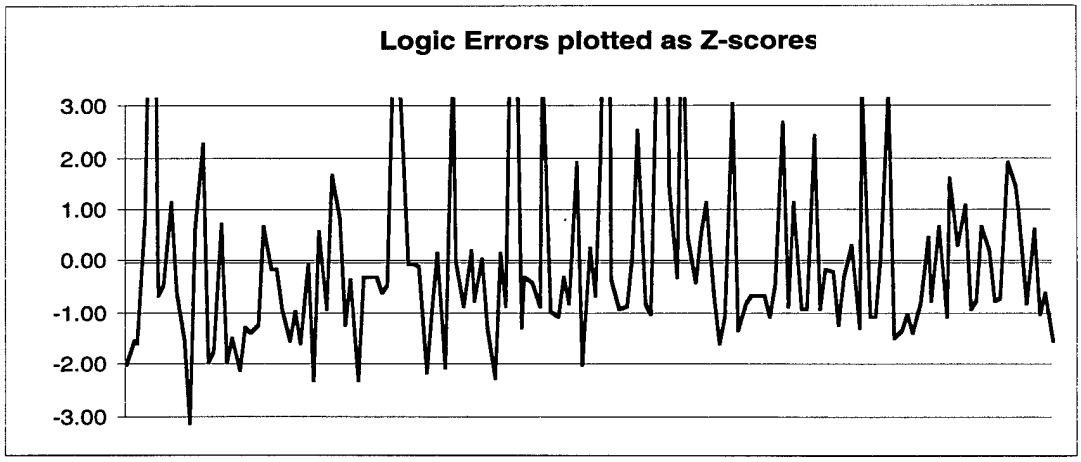


Figure C-8: Short-Run Attribute Control Chart for Logic Errors

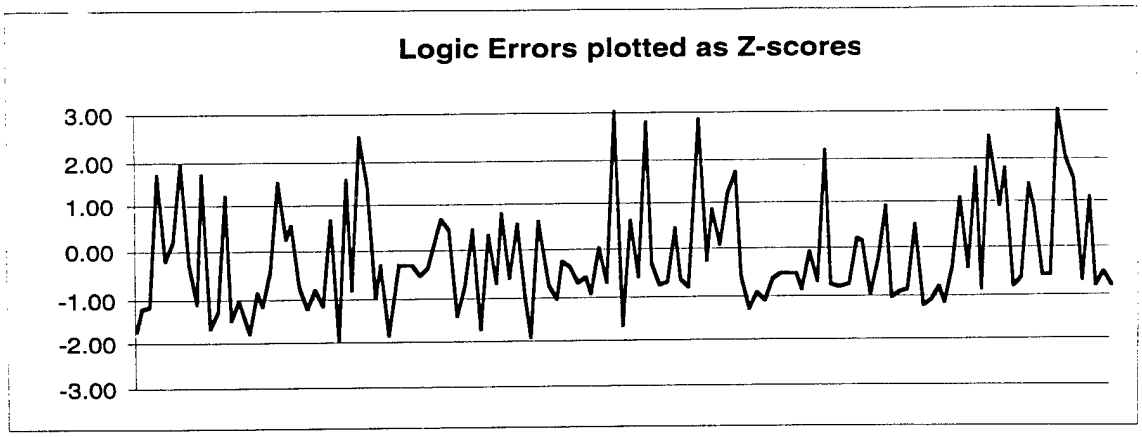


Figure C-9: Short-Run Attribute Control Chart for Logic Errors After Eliminating Reviews with Special Causes

Figure C-9 shows that the project's process is in control because all the points are randomly distributed within the upper and lower control limits (+3.00 and -3.00).

We then calculated the number of "escaped" defects. This is a metric that can be readily computed from summary defect data. Our definition of an escaped defect is one that should have been found in an inspection during an earlier software development phase. This means that requirements defects found during design or code inspections are considered escapes, and design defects found during code inspections are also escapes. The chart below shows the total number of defects found during each software development phase where peer reviews were held. Of the total 3769 defects found, 123 defects (those below the diagonal [9 + 1 + 113]) were classified as "escapes". The ratio of escaped defects to total defects found was computed as 3.26%.

Defect Type

Peer Review Where Defect Found	Requirements	Design	Code
Requirements	**0**		
Design	**9**	**1711**	
Code	**1**	**113**	**1935**

Figure C-10: Total Number of Defects Found, with Escaped Defects Shown Below the Shaded Line

C.8 Round 3 Wrap-Up

We consulted with our VR expert again to review the analysis. As our understanding of the problem increased, we became concerned about the validity of the statistical methods we had employed. The assumption associated with u charts is that the number of defects found in a given inspection follow a Poisson distribution. We wanted to know if tests on the actual distribution were necessary. He told us that the attribute of our process that makes it applicable to u charts was the fact that there are many opportunities for defects with a very small probability that a given line will have one or more defects. This definitely applies because the measured defects per line is about 0.01 when all of the defects are counted. It is even smaller when the number of defect categories is restricted. This meeting gave us confidence that we were on firm theoretical grounds.

C.9 Conclusions

Our team felt they obtained some valuable information from this exercise. We determined that defect data from peer reviews can be used to highlight areas for process improvement. We also found that if we assume a stable peer review process, we can determine a range for the number of defects we expect to find during a peer review, based on the size of the review package. This concept can be best applied when the same software group has gone through a complete software development life cycle and will be repeating the process again on another development effort. We also found that the project was very interested in the reports we developed. They initiated changes in both their software development process and their peer review process based on the information we gleaned from their data, adding the use of a code standards checker prior to the peer review and combining some of the defect categories to reduce the time spent reviewing the material. Another important aspect of this study is that we showed that an organizational group, the SEPG, can develop a process that supports process improvements on individual projects without much effort on the part of project personnel.

C.10 Future Plans

The metrics team has planned several activities as a follow-up to this effort. Our plans include development of a standard peer review data analysis process and extension of the process to other types of software defects, such as those found during testing and after delivery. First, we plan to analyze inspection data from the second project to determine if the analysis techniques we used can be used on projects developing different types of software applications. If we perform a similar analysis on a set of data from the second software project and can recommend improvements to their processes, we will develop a documented process for application to other MDA software projects. To assist the transition of the process to our software organization, we will investigate tools that can support the collection and analysis of the defect data. Members of the VR group have offered to develop an application for analyzing software peer review defects using Statistical Analysis System

(SAS). We are also investigating automation of the defect data entry, collection, and analysis process through the use of Excel macros.

After development of a peer review data analysis process we will begin looking at software errors found in the later stages of the software development process. We plan to develop an enterprise-wide approach to collecting software errors during software and system-level testing and after delivery. We then plan to develop techniques that will help us analyze software errors detected throughout the software development process. We can then use this process to measure the impacts of process improvements on product quality.

References

[Austin 93] Austin, Robert D. & Paulish, Daniel J., *A Survey of Commonly Applied Methods for Software Improvement* (CMU/SEI-93-TR-27, ADA278595). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, February 1993.

[Bhandari 93] Bhandari, I. et al. "A Case Study of Software Process Improvement During Development." *IEEE Transactions on Software Engineering 19*, 12 (December 1993): 1157–1170.

[Basili 94] Basili, Victor R. & Green, Scott. "Software Process Evolution at the SEL." *IEEE Software 11*, 4 (July 1994): 58–66.

[Beauregard 92] Beauregard, Michael R.; Mikulak, Raymond J.; & Olson, Barbara A. *A Practical Guide to Statistical Quality Improvement—Opening Up the Statistical Toolbox.* New York, N.Y.: Van Nostrand Reinhold, 1992.

[Brassard 88] Brassard, Michael, ed. *The Memory Jogger: A Pocket Guide of Tools for Continuous Improvement.* 2nd ed. Methuen, Mass.: GOAL/QPC, 1988.

[Brassard 89] Brassard, Michael. *The Memory Jogger+™: Featuring the Seven Management and Planning Tools.* Methuen, Mass.: GOAL/QPC, 1989.

[Brodman 95] Brodman, Judith G. & Johnson, Donna L. "Return on Investment (ROI) from Software Process Improvement as Measured by US Industry." *Software Process: Improvement and Practice 1*, Pilot Issue (August 1995): 35–47.

[Burr 53] Burr, Irving W. *Engineering Statistics and Quality.* New York, N.Y.: McGraw-Hill, 1953.

[Burr 67] Burr, Irving W. "The Effect of Non-Normality on Constants for X-bar and R Charts." *Industrial Quality Control* (1967): 563–568.

[Card 90] Card, David N. & Glass, Robert L. *Measuring Software Design Quality.* Englewood Cliffs, N.J.: Prentice Hall, 1990.

[Card 95] Card, David N. "The RAD Fad: Is Timing Really Everything?" *IEEE Software 12*, 5 (September 1995): 19–22.

[Champ 87] Champ, Charles. W. & Woodall, William. H. "Exact Results for Shewhart Control Charts with Supplementary Runs Rules." *Technometrics 29*, 4 (November 1987): 393–399.

[Chillarege 92] Chillarege, R. et al. "Orthogonal Defect Classification—A Concept for In-Process Measurements." *IEEE Transactions on Software Engineering 18*, 11 (November 1992): 943–956.

[Chillarege 96] Chillarege, Ram. Ch. 9, "Orthogonal Defect Classification," 359–400. *Handbook of Software Reliability Engineering.* Michael R. Lyu, ed. Los Alamitos, Calif.: IEEE Computer Society Press, 1996. (Also available from McGraw-Hill Book Company, New York, N.Y.).

[Deming 75] Deming, W. Edwards. "On Probability As a Basis For Action." *The American Statistician 29*, No. 4 (November 1975): 146–152.

[Deming 80] Deming, W. Edwards. "Dedication" to the reprint of Walter A. Shewhart's *Economic Control of Quality of Manufactured Product.* Milwaukee, Wisc.: American Society of Quality Control, 1980.

[Deming 86a] Deming, W. Edwards. *Out of the Crisis.* Cambridge, Mass.: Massachusetts Institute of Technology, Center for Advanced Engineering, 1986.

[Deming 86b] Deming, W. Edwards. "Foreword" to the reprint of Walter A. Shewhart's *Statistical Method from the Viewpoint of Quality Control.* Mineola, N.Y.: Dover Publications, Inc., 1986.

[Deming 93] Deming, W. Edwards. *The New Economics for Industry, Government, Education.* Cambridge, Mass.: Massachusetts Institute of Technology, Center for Advanced Engineering, 1993.

[Dillman 83] Dillman, D. A. "Mail and Other Self-Administered Questionnaires." *Handbook of Survey Research.* P. H. Rossi, J. D. Wright, & A. B. Anderson, ed. New York, N.Y.: Academic Press, 1983.

[Duncan 74] Duncan, Acheson J. *Quality Control and Industrial Statistics.* Homewood, Ill.: Richard D, Irwin, Inc., 1974.

[Fernie 96] Fernie, J. Donald. "The Mysterious Gamma-Ray Bursters." *American Scientist 84*, 5 (September-October 1996): 434–436.

[Florac 92] Florac, William A. et al. *Software Quality Measurement: A Framework for Counting Problems and Defects* (CMU/SEI-92-TR-22, ADA258556). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, September 1992.

[Goethert 92] Goethert, Wolfhart B.; Bailey, Elizabeth K.; & Busby, Mary B. *Software Effort and Schedule Measurement: A Framework for Counting Staff-Hours and Reporting Schedule Information* (CMU/SEI-92-TR-21, ADA258279). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, September 1992.

[Grady 87] Grady, Robert B. & Caswell, Deborah L. *Software Metrics: Establishing a Company-Wide Program*. Englewood Cliffs, N.J.: Prentice Hall, 1987.

[Grady 92] Grady, Robert B. *Practical Software Metrics for Project Management and Process Improvement.* Englewood Cliffs, N.J.: Prentice Hall, 1992.

[Grady 94} Grady, Robert B. & Van Slack, Tom. "Key Lessons In Achieving Widespread Inspection Use." *IEEE Software 11*, 4 (July 1994): 46–57.

[Grant 96] Grant, Eugene L. & Leavenworth, Richard S. *Statistical Quality Control,* 7th ed. New York, N.Y.: McGraw-Hill, 1996.

[Grove 96] Grove, Andy. *Only the Paranoid Survive.* New York, N.Y.: Doubleday, 1996.

[Hahn 70] Hahn, Gerald J. "Statistical Intervals for a Normal Population." *Journal of Quality Technology 2*, Nos. 3 & 4 (July 1970, October 1970): 115–125, 195–206.

[Hahn 91] Hahn, Gerald J. & Meeker, William Q. *Statistical Intervals: A Guide for Practitioners.* New York, N.Y.: John Wiley & Sons, Inc., 1991.

[Hahn 93] Hahn, Gerald J. & Meeker, William Q. "Assumptions for Statistical Inference." *The American Statistician 47*, 1 (February 1993): 1–11.

[Harter 60] Harter, H. L. "Tables of Range and Studentized Range." *Annals of Mathematical Statistics 31* (1960): 1122–1147.

[Henry 92] Henry, Joel & Blasewitz, Bob. "Process Definition: Theory and Reality." *IEEE Software 9,* 6 (November 1992): 103–105.

[Hillier 69] Hillier, Frederick S. "$\bar{X}$ and R-Chart Control Limits Based on a Small Number of Subgroups." *Journal of Quality Technology 1*, 1 (January 1969): 17–26.

[Hoyer 96] Hoyer, Robert A. & Ellis, Wayne C. "A Graphical Exploration of SPC." *Quality Progress 29*, Nos. 5 & 6 (May 1996): 65–73, (June 1996): 57–64.

[Humphrey 89] Humphrey, Watts S. *Managing the Software Process.* Reading, Mass.: Addison-Wesley, 1989.

[Humphrey 95] Humphrey, Watts S. *A Discipline for Software Engineering.* Reading, Mass.: Addison-Wesley, 1995.

[Ishikawa 86] Ishikawa, Kaoru. *Guide to Quality Control,* Second Revised Edition. White Plains, N.Y.: UNIPUB – Kraus International Publications, 1986.

[JLC 96] Joint Logistics Commanders Joint Group on Systems Engineering. *Practical Software Measurement: A Guide to Objective Program Insight, Version 2.1.* Newport, R.I.: Naval Undersea Warfare Center, March 1996.

[Juran 88] Juran, J. M. & Gryna, Frank M., ed. *Juran's Quality Control Handbook, Fourth Edition.* New York, N.Y.: McGraw-Hill Book Company, 1988.

[Kaplan 92] Kaplan, R. S. & Norton, D. P. "The Balance Scorecard—Measures that Drive Performance." *Harvard Business Review* (Jan–Feb 1992): 71–79.

[Kenett 91] Kenett, Ron S. "Two Methods for Comparing Pareto Charts." *Journal of Quality Technology 23*, 1 (January 1991): 27–31.

[Kirchner 59] Kirchner, Paul. "Measurements and Management Decisions," 64–82. *Measurement: Definitions and Theories*, C. West Churchman & Philburn Ratoosh, ed. New York, N.Y.: John Wiley & Sons, Inc., 1959.

[Lynch 95] Lynch, Richard L. & Cross, Kelvin F. *Measure Up!* Cambridge, Mass: Blackwell Publishers, Basil Blackwell Inc., 1995.

[Montgomery 96] Montgomery, Douglas. C. *Introduction to Statistical Quality Control,* 3rd ed. New York, N.Y.: John Wiley & Sons, 1996.

[Mood 74] Mood, Alexander M.; Graybill, Franklin A.; & Boes, Duane C. *Introduction to the Theory of Statistics,* 3rd ed. New York, N.Y.: McGraw-Hill Book Company, 1974.

[Natrella 66] Natrella, Mary G. *Experimental Statistics (National Bureau of Standards Handbook 91).* Washington, DC: U.S. Government Printing Office, 1966.

[Ott 90] Ott, Ellis R. & Schilling, Edward G. *Process Quality Control,* 2nd ed. New York, N.Y.: McGraw-Hill Publishing Company, 1990.

[Pall 87] Pall, Gabriel A. *Quality Process Management.* Englewood Cliffs, N.J.: Prentice Hall, 1987.

[Park 92] Park, Robert E. et al. *Software Size Measurement: A Framework for Counting Source Statements* (CMU/SEI-92-TR-20, ADA258304). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, September 1992.

[Park 95] Park, Robert E. *Checklists and Criteria for Evaluating the Cost and Schedule Estimating Capabilities of Software Organizations* (CMU/SEI-95-SR-05, ADA293299). Pittsburgh, Pa: Software Engineering Institute, Carnegie Mellon University, January 1995.

[Park 96a] Park, Robert E.; Goethert, Wolfhart B.; & Florac, William A. *Goal-Driven Software Measurement—A Guidebook* (CMU/SEI-96-HB-002, ADA313946). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, July 1996.

[Park 96b] Park, Robert E. "Assessing an Organization's Estimating Capabilities." *American Programmer 9*, 7 (July 1996): 42–49.

[Paulk 93a] Paulk, Mark C. et al. *Capability Maturity Model for Software, Version 1.1* (CMU/SEI-93-TR-24, ADA263403). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, February 1993.

[Paulk 93b] Paulk, Mark C. et al. *Key Practices of the Capability Maturity Model, Version 1.1* (CMU/SEI-93-TR-25, ADA263432). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, February 1993.

[Paulk 95] Paulk, Mark C.; Weber, Charles V.; Curtis, Bill; Chrissis, Mary Beth; et al. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Reading, Mass.: Addison-Wesley, 1995.

[Perry 94] Perry, Dewayne E. & Votta, Lawrence G. "People, Organizations, and Process Improvement." *IEEE Software 11*, 4 (July 1994): 36–45.

[Proschan 60] Proschan, Frank & Savage, I. R. "Starting a Control Chart: The Effect of Number and Size of Samples on the Level of Significance at the Start of a Control Chart for Sample Means." *Industrial Quality Control* (September 1960): 12–13.

[Pyzdek 90] Pyzdek, Thomas. *Pyzdek's Guide to SPC: Volume One, Fundamentals*. Tucson, Ariz.: Quality Publishing, Inc., 1990.

[Pyzdek 92] Pyzdek, Thomas. *Pyzdek's Guide to SPC: Volume Two, Applications and Special Topics*. Tucson, Ariz.: Quality Publishing, Inc., 1992.

[Quesenberry 93] Quesenberry, Charles P. "The Effect of Sample Size on Estimated Limits for $\bar{X}$ and X Control Charts." *Journal of Quality Technology 25*, 4 (October 1993): 237–247.

[Rigdon 94] Rigdon, Steven E.; Cruthis, Emma N.; & Champ, Charles W. "Design Strategies for individuals and Moving Range Control Charts." *Journal of Quality Technology 26*, 4 (October 1994): 274–287.

[Roberts 96] Roberts, Mark A. "Experiences in Analyzing Software Inspection Data." *Proceedings of the 1996 SEPG Conference: Tuesday, May 21 Papers*. Atlantic City, N.J., 20–23 May 1996. Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, May 1996.

[Roes 93] Roes, Kit C. B.; Does, Ronald J. M. M.; & Schurink, Yvonne. "Shewhart-Type Control Charts for Individual Observations." *Journal of Quality Technology 25*, 3 (July 1993): 188–198.

[Rombach 89] Rombach, H. Dieter & Ulery, Bradford T. "Improving Software Maintenance Through Measurement." *Proceedings of the IEEE 77*, 4 (April 1989): 581–595.

[Scheuer 90] Scheuer, E. M.. "Let's Teach More About Prediction," 133–137. *Proceedings of the Statistical Education Section, American Statistical Association.* Washington, D.C.: American Statistical Association, 1990.

[Senge 94] Senge, Peter M. *The Fifth Discipline Fieldbook.* New York, N.Y.: Doubleday, 1994.

[Shepperd 93] Shepperd, Martin & Ince, Darrel. *Derivation and Validation of Software Metrics.* Oxford, England: Clarendon Press, 1993.

[Shewhart 31] Shewhart, Walter A. *Economic Control of Quality of Manufactured Product.* New York, N.Y.: Van Nostrand, 1931. (Reprinted in Milwaukee, Wisc.: American Society of Quality Control, 1980.)

[Shewhart 39] Shewhart, Walter A. *Statistical Method from the Viewpoint of Quality Control.* Washington, DC: Graduate School of the Department of Agriculture, 1939. (Reprinted in Mineola, N.Y.: Dover Publications, Inc., 1986.)

[Shewhart 43] Shewhart, Walter A. "Statistical Control in Applied Science." *Transactions of the A.S.M.E. 65* (April 1943): 222–225.

[Vardeman 92] Vardeman, Stephen B. "What About the Other Intervals?." *The American Statistician 46*, 3 (August 1992): 193–197.

[Wadsworth 86] Wadsworth, Harrison M.; Stephens, Kenneth S.; & Godfrey, A. Blanton. *Modern Methods for Quality Control and Improvement.* New York, N.Y.: John Wiley & Sons, 1986.

[Western Electric 58] Western Electric Co., Inc. *Statistical Quality Control Handbook.* Indianapolis, Indiana: AT&T Technologies, 1958.

[Wheeler 89] Wheeler, Donald J. & Lyday, Richard W. *Evaluating the Measurement Process.* Knoxville, Tenn.: SPC Press, 1989.

[Wheeler 92] Wheeler, Donald J. & Chambers, David S. *Understanding Statistical Process Control.* Knoxville, Tenn.: SPC Press, 1992.

[Wheeler 93] Wheeler, Donald J. *Understanding Variation: The Key to Managing Chaos.* Knoxville, Tenn.: SPC Press, 1993.

[Wheeler 95] Wheeler, Donald J. *Advanced Topics in Statistical Process Control.* Knoxville, Tenn.: SPC Press, 1995.

Index

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (LEAVE BLANK)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	April 1997	Final

4. TITLE AND SUBTITLE

Practical Software Measurement: Measuring for Process Management and Improvement

5. FUNDING NUMBERS

C — F19628-95-C-0003

6. AUTHOR(S)

William A. Florac, Robert E. Park, Anita D. Carleton

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

8. PERFORMING ORGANIZATION REPORT NUMBER

CMU/SEI-97-HB-003

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

HQ ESC/AXS
5 Eglin Street
Hanscom AFB, MA 01731-2116

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12.A DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified/Unlimited, DTIC, NTIS

12.B DISTRIBUTION CODE

13. ABSTRACT (MAXIMUM 200 WORDS)

This guidebook shows how well-established principles and methods for evaluating and controlling process performance can be applied in software settings to help achieve an organization's business and technical goals. Although the concepts that are illustrated are often applicable to individual projects, the primary focus is on the enduring issues that enable organizations to improve not just today's performance, but the long-term success and profitability of their business and technical endeavors.

14. SUBJECT TERMS

process improvement, process management, process performance, quality improvement, software measurement, software process, statistical process control, SPC, statistical quality control

15. NUMBER OF PAGES

240

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102